Type of Query	SQL
Create table	CREATE TABLE table_name (A1D1, AnDn, IC1, ICn)
Delete relation	DROP TABLE table_name
Add column	ALTER TABLE table_name ADD COLUMN column_name
Add record	INSERT INTO table_name(A1,An) VALUES (V1,Vn)
Delete record based on condition	DELETE FROM table_name WHERE condition
Delete all records	DELETE FROM table_name
Update records based on condition	UPDATE table_name SET Ai = Vi WHERE Aj = Vj
Select records based on condition	SELECT A1, An FROM table_name WHERE condition
Select without duplicates	SELECT DISTINCT A1, An FROM table_name WHERE condition
Select with duplicates	SELECT ALL A1, An FROM table_name WHERE condition
Select all attributes	SELECT * FROM table_name WHERE condition
Selection resulting in a 1 by 1 relation of given value	SELECT 'value'
Selection resulting in a n-row relation containing given value	SELECT 'value' FROM table_name
Rename attribute name	SELECT A1 AS name
Selection with arithmetic expression	SELECT A1 (+,-,*,/) FROM table_name
Logical operators	AND, NOT, OR
Comparisons	=, <>, <, >, <=, >=
Select Cartesian product	SELECT * FROM table1, tablen
Rename table name	SELECT * FROM table AS name
Left outer join	SELECT * FROM t1 LEFT OUTER JOIN t2 ON t1.a = t2.a
Right outer join	SELECT* FROM t1 RIGHT OUTER JOIN t2 ON t1.a = t2.a
Full outer join	SELECT * FROM t1 FULL OUTER JOIN t2 ON t1.a = t2.a
Inner join	SELECT * FROM t1 INNER JOIN t2 ON t1.a = t2.a
String matching	SELECT * FROM table WHERE a LIKE "_abc%"
Range query	SELECT * FROM table WHERE a BETWEEN x AND y
Set operations	UNION, INTERSECT, EXCEPT
Select where attribute is null	SELECT * FROM table WHERE a IS NULL
Select record where attribute matches one of multiple values	SELECT * FROM table WHERE a IN (v1, vn)
Aggregate values	AVG, MIN, MAX, SUM, COUNT
Select average value of an attribute	SELECT AVG(a) FROM table
Group by values within an attribute	SELECT * FROM table GROUP BY a
Group by values within an attribute and specify condition	SELECT * FROM table GROUP BY a HAVING condition
Data types	CHAR(n), VARCHAR(n), INT, SMALLINT, NUMERIC(p,d), REAL, FLOAT(n)
Specify primary key upon creation	CREATE TABLE table_name (A1D1, AnDn, PRIMARY KEY (Ai))
Specify foreign key upon creation	CREATE TABLE table_name (A1D1, AnDn, FOREIGN KEY Ai REFERENCES table2)
Referential integrity	FOREIGN KEY Ai REFERENCES table2 ON DELETE/UPDATE NO ACTION/CASCADE/SET DEFAULT
Check clause	CREATE TABLE table_name (A1D1 CHECK condition)

Relational algebra symbols:

$\cup$ union, $\cap$ intersection, - difference, σ select, π projection, $\times$ cartesian product, $\bowtie$ natural join, ρ rename, δ duplicate elimination, γ grouping and aggregation, τ sorting.

Relational algebra examples:

Q: Find the titles of courses in the Comp. Sci. department that have 3 credits. A: $\pi_{title}(\sigma_{dept\_name='Comp.Sci.' \wedge credits=3}(course))$

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

A: $\pi_{ID}(takes \bowtie \pi_{course\_id}(teaches \bowtie \sigma_{name='Einstein'}(instructor)))$

Q: Find the highest salary of any instructor.

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

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

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

Description	Python code
Create DataFrame from pandas series	data = {col1:[11, 12, 13], {col2:[21,22,23]} df = pd.DataFrame(data)
Create DataFrame from data matrix	data = [[11, 21,31],[12,22,32],[13,23,33]] cols = [col1,col2,col3] df = pd.DataFrame(data, columns = cols)
Create DataFrame from CSV file	df = pd.read_csv('file.csv')
Create DataFrame from Excel file	workbook = pd.ExcelFile('file.xlsx') df = workbook.parse(sheet_name)
Get index and datatypes from DataFrame	df.info()
Get first n rows from DataFrame	df.head(n)
Get last n rows from DataFrame	df.tail(n)
Get summary stats for DataFrame	df.describe()
Extract row n by index	df.iloc[n,:]
Extract column n by index	df.iloc[:,n]
Extract column by name	df.col_name or df["col_name"]
Extract rows based on condition	df.loc[df.col==value, [C1,Cn]]
Display number of columns	print(len(df.columns))
Display number of rows	print(len(df))
Find number of non-null values for each column	df.count()
Display number of distinct values for column	print(len(df.unique()))
Date type of column	df['column'].dtype
Aggregate query	df['column'].mean() (.min(), .max(), .median(), .std())
Concatenate row-wise	union_df = pd.concat([df1, df2])
Concatenate column-wise	union_df = pd.concat([df1, df2], axis=1)
Join operation	merge_df = pd.merge(df1, df2, on='col') merge_df = pd.merge(df1, df2, left_on='df1_col', right_on='df2_col')
Find column name by index	df.columns(n)
Find number of missing values in col	df.col.isnull().sum()

Functional dependencies	
Reflexivity Axiom	If X $\subset Y, thenY \longmapsto X$
Augmentation Axiom	If X $\longmapsto Y, thenAX \longmapsto AY$
Transivity Axiom	If X $\longmapsto YandY \longmapsto W, thenX \longmapsto W$
First Normal Form (1NF)	Relation contains only atomic values (no multiple values in one cell)
Second Normal Form (2NF)	Relation is in 1NF and all non-key attributes are fully functional dependent on the primary key
Third Normal Form (3NF)	Relation is in 2NF and there is no transitive dependency
Boyce Codd Normal Forme (BCNF)	Relation is in 3NF and for functional dependency X $\longmapsto Y, XmustbeaSuperKey$

String similarity	
Levenshtein distance	Minimum number of single-character edits required to change one string into the other
Jaro Similarity	$JaroSim(S1,S2) = \frac{1}{3}(\frac{C}{S1} + \frac{C}{S2} + (\frac{C-T}{C})$, C=number of common characters, T=number of transpositions/2, S1 and S2 = respective lengths of the strings
Jaro-Winkler Similarity	$Jw(S1,S2) = JaroSim * P * L * (1 - JaroSim)$,P=Scaling factor (0.1 default), L=length of common prefix up to maximum 4
Soundex	Phonetic algorithm that indexes names by their sounds when pronounced in English

Set similarity	
Jaccard Similarity	Finds the similarity between sets by dividing the intersection over the union, $J_{sim}(C_1,C_2) = \frac{\lvert C_1 \cap C_2 \rvert}{\lvert C_1 \cup C_2 \rvert}$
Jaccard Distance	Finds the dissimilarity (or distance) between sets by dividing de difference over the union, $J_{sim}(C_1,C_2) = \frac{\lvert C_1 - C_2 \rvert}{\lvert C_1 \cup C_2 \rvert}$
Jaccard Bag Similarity	Counts the repitition of elements (The similarity between {a,a,a,b} and {a,a,b,b,c} = 3/9 = 1/3)
Sørensen Coefficient	Finds the similarity between sets by dividing two times the intersection over the total number of elements, $CC(C_1,C_2) = \frac{2 * \lvert C_1 \cap C_2 \rvert}{\lvert C_1 \rvert + \lvert C_2 \rvert}$
Overlap Coefficient	Finds the similarity between sets by dividing the intersection over the amount of elements of the set with the least amount of elements, $OC(C_1,C_2) = \frac{\lvert C_1 \cap C_2 \rvert}{MIN(\lvert C_1 \rvert, \lvert C_2 \rvert)}$
Tversky Index	A generalized form of Jaccard and Sørensen $S(C_1,C_2) = \frac{\lvert C_1 \cap C_2 \rvert}{\lvert C_1 \cap C_2 \rvert + \alpha \lvert C_1 - C_2 \rvert + \beta \lvert C_2 - C_1 \rvert}$, Jaccard similarity: $\alpha = \beta = 1$, Sørensen: $\alpha = \beta = 0.5$

Hashing example
h3 = 2x+4 mod 5
h4 = 3x-1 mod 5

Element	S1	S2	S3	S4	h3	h4
0	0	1	0	1	4	4
1	0	1	0	0	1	2
2	1	0	0	1	3	0
3	0	0	1	0	0	3
4	0	0	1	1	2	1
5	1	0	0	0	4	4
h3	3	1	0	2		
h4	0	2	1	0		

Data preparation	
Handling Missing Data	Ignore, Fill in (manually, automatically (global constant, mean, etc.))
Handling Noisy Data	Binning, Regression, Clustering, Combined computer and human inspection
Equidepth bins	Each bin has same amount of elements and different range
Equidistance bins	Each bin has same range and different amount of elements

Outlier detection	
Outlier	Observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism
Types of outliers	Global, contextual, collective
Outlier Detection Approaches	Statistical approach, Distance-based approach, Density-based approach, Model-Based approach
Local Outlier Factor (LOF)	Quantifies the local density of a data point, with the use of a neighborhood of size k

Data normalization method	Formula
Min-max normalization	$v' = \frac{v-min}{max-min}(new\_max - new\_min) + new\_min$
Z-score normalization	$v' = \frac{v-\mu}{\sigma}$
Decimal scaling normalization	$v' = \frac{v}{10^j}$ where j is the smallest integer such that $max(\lvert v' \rvert) < 1$

Data reduction	
Principal Component Analysis (PCA)	Dimensionality reduction by finding a projection that captures the largest amount of variation in data. Works for numeric data only.
Attribute Subset Selection	Dimensionality reduction by removing redundant attributes and removing irrelevant attributes
Model-Based Data Reduction	Data reduction by modeling to a straight line (linear regression, multiple regression, log-linear model)
Histogram for Data Reduction	Divide data into buckets and store the average for each bucket
Clustering-Based Data Reduction	Partition dataset into clusters based on similarity, and store cluster representation
Sampling-Based Data Reduction	Obtain a small sample to represent the whole dataset
Simple random sampling	Equal probability of selecting any particular item
Sampling without replacement	Once an object is selected, it is removed from the population
Sampling with replacement	A selected object is not removed from the population
Stratified sampling / Cluster sampling	Partition the dataset, and draw samples from each partition. Used in conjunction with skewed data.
Concept Hierarchy Generation	Each level of the hierarchy represents a concept that is more general than the level below it. Some hierarchies can be automatically generated based on the analysis of the number of distinct values per attribute in the data set

Cloud computing	
5Vs of Big Data	Big Data Characteristics: Volume, Variety, Velocity, Veracity, Value
Cloud Computing	A compilation of technologies, packaged within a infrastructure paradigm that offers improved scalability, elasticity, business agility, faster startup time, reduced management costs and just-in-time availability of resources
Characteristic Pillars of Cloud Computing	Efficiency, Accessibility, Ultra-Reliability, On-demand, Elasticity, Scalability, Sustainability
Cloud Deployment Models	Public Cloud, Private Cloud, Hybrid Cloud, Community Cloud
Cloud delivery models	Infrastructure as a Service, Platform as a Service, Software as a Service

Visualtization & EDA	
Grammar of Graphics	Map raw data to: Aesthetics, Geometric objects, Scales, Facets. Additionaly can apply: Statistical transformation, Coordinate system
Gestalt principles of relatedness	Proximity, Smilarity, Connection, Continuity, Closure, Figure and ground, Common fate
Tufte data to ink ratio principle	The ratio between the amount of information the plot shows and the amount of "ink" being used, principle is to maximize this ratio (within reason)
Types of analysis	Exploratory Analysis and Confirmatory Analysis
Exploratory Analysis	Generate new insights, Create new hypotheses, Analysis may depend on data
Confirmatory Analysis	Test theory, A priori hypotheses, Analysis predefined
Tukey's aproach to EDA	Look at center and spread. Find comparisons. "Straightening and flattening": use logarithms and other transforms, use models and residuals
Peng's EDA checklist	1. Formulate your question, 2. Read in your data, 3. Check the packaging, 4. Look at the top and bottom of your data, 5. Check your "n"s, 6. Validate with at least one external source, 7. Try the easy solution first, 8. Challenge your solution, 9. Follow up

R	
Plot	ggplot(data = df) + geom_point(mapping = aes(x=a1, y=a2, color=a3, size=a4))
Subplots	+ facet_wrap(∼a1 nrow=n)
Head	df %>% head
Check n	df %>% group_by (a1) %>% summarize(n())
Filter	df %>% filter(a1 == value)
Create df	df <- tibble(col1 = data1, coln = datan)
Create formula	formula <- outcome ∼pred1 + pred2
Create linear model object	lm_ses <- lm(formula = medv ∼lstat, data = data)
Predict from model	y_pred <- predict(lm_ses)
Read CSV	df <- read_csv('file.csv')
Other geoms	geom_density, geom_bar, geom_line, geom_histogram, geom_rug, geom_boxplot, geom_col, geom_smooth, geom_ribbon

Term	Definition
OSEMN framework	Obtain, Scrub, Explore, Model, iNterpret
CRISP-DM	Cross Industry Standard Process for Data Mining
KDD Process	Knowledge Discovery in Databases
Data Wrangling	An iterative process to convert the raw data into a more understandable format. Discovering, Structuring, Cleaning & Transformation, Enriching, Validating, Publishing
Data Formats	Structured Data, Unstructured Data, Semistructred Data
Metadata	Data about data
Data Models	Mathematical representation of the data. Collection of tools describing relationships, semantics, constraints, operations.
Main Components of a Data Model	Structures, Constraints, Operations
Parts of the Relational Model	Instance, Schema
Types of keys	Superkey, Candidate key, Primary key, Foreign key
Constraints	Guard against accidental damage to the data, Ensure that authorized changes to the data don't violate consistency
Drawback of file system as data storage	Data redundancy and inconsistency, Difficulty in accessing data, Data isolation, Integrity problems, Atomicity of updates, Concurrent access by multiple users, Security problems. Database systems offer solutions to all these problems.
Databases	Traditionally: systems that contain records about real world entities Today: covers all the largest sources of data (web search, data mining, etc.)
Database levels of abstraction	Physical level, Logical level, View level
Data Definition Language (DDL)	Specification notation for defining the database schema
Data Manipulation Language (DML)	Language for accessing and manipulating the data. Also known as query language
Supervised learning	Learn a model from labeled training data, then make predictions
Unsupervised learning	Explore the structure of the data to extract meaningful information
Reinforcement Learning	Develop an agent that improves its performance based on interactions with the environment