

Databases for Data Science

Module II

Module II

- ✓ Structured Query Language (SQL):
- ✓ Basic Statistics, Data Munging, Filtering, Joins,
- ✓ Aggregation, Window Functions, Ordered Data, preparing No-SQL: Document Databases,
- ✓ Wide-column Databases and Graphical Databases

- ✓ Efficiency and effectiveness of data processing and analysis

The choice of a database

- ✓ Volume, structure, and query complexity.
- ✓ Consider factors such as data structure, scalability, query performance, ease of use, and the specific requirements of your analysis. Additionally, cloud-based database services, such as Amazon RDS, Google Cloud Spanner, and Microsoft Azure Cosmos DB

Relational Databases:

- ✓ **MySQL:** An open-source relational database management system (RDBMS) widely used for web applications and data-driven projects.
- ✓ **PostgreSQL:** An advanced open-source RDBMS known for its extensibility and support for complex queries.
- ✓ **SQLite:** A lightweight, serverless, and self-contained database engine suitable for embedded systems and small-scale applications.

NoSQL Databases:

- ✓ **MongoDB:** Document-oriented NoSQL database, stores data in JSON-like BSON documents, Scalable and flexible data storage.
- ✓ **Cassandra:** Distributed NoSQL database, designed for handling large amounts of data across multiple commodity servers, known for its high availability and fault tolerance.
- ✓ **CouchDB:** NoSQL database uses a document-oriented model, providing a scalable and distributed architecture.

Columnar Databases:

- ✓ **Apache HBase:** A distributed, scalable, and column-oriented database that is part of the Apache Hadoop project, suitable for handling large datasets.
- ✓ **Google Bigtable:** A fully managed, highly scalable, and columnar database service designed for large analytical and operational workloads.

Graph Databases:

- ✓ **Neo4j:** A popular graph database that is efficient in managing and querying highly interconnected data, making it suitable for applications involving complex relationships.
- ✓ **Amazon Neptune:** A fully managed graph database service compatible with both property graph and RDF graph models, designed for high-performance graph queries.

Time Series Databases:

- ✓ **InfluxDB:** Scalable and high-performance time-series database designed for handling large volumes of time stamped data.
- ✓ **Prometheus:** Open-source monitoring and alerting toolkit with a built-in time-series database, commonly used for collecting metrics from various systems.

Document Stores:

- ✓ **Elastic search:** Initially designed for full-text search, Used as a distributed document store for indexing and searching large volumes of data.
- ✓ **Amazon DynamoDB:** Fully managed NoSQL database service provided by AWS, suitable for document and key-value store use cases.

NewSQL Databases:

- ✓ **CockroachDB:** A distributed SQL database that provides ACID transactions and scalability, designed to scale horizontally while maintaining strong consistency.
- ✓ **Google Spanner:** A globally distributed, horizontally scalable, and strongly consistent NewSQL database designed for global transaction consistency.

Spatial Databases:

- ✓ **PostGIS:** An extension for PostgreSQL that adds support for spatial data and geospatial analysis.
- ✓ **MongoDB with GeoJSON:** MongoDB supports geospatial queries and indexing through GeoJSON objects.

RDF Stores:

- ✓ **Apache Jena:** A Java framework for building semantic web and linked data applications, including support for RDF storage and querying.
- ✓ **Stardog:** A knowledge graph platform that supports RDF and SPARQL for building and querying semantic data.(query information from databases or any data source that can be mapped to RDF)
(triplestore or RDF store is a purpose-built database for the storage and retrieval of triples through semantic queries)

Basic Statistics

Descriptive Statistics:

- **Mean (Average):** The sum of all values divided by the number of values in a dataset. It represents the central tendency of the data.
- **Median:** The middle value in a sorted dataset. It is less sensitive to extreme values than the mean and provides insight into the data's distribution.
- **Mode:** The most frequently occurring value in a dataset.

Measures of Dispersion:

- **Range:** The difference between the maximum and minimum values in a dataset, providing a simple measure of spread.
- **Variance:** The average of the squared differences from the mean. It measures the overall variability of the data.
- **Standard Deviation:** The square root of the variance. It provides a more interpretable measure of the spread of the data.

- **Percentiles and Quartiles:**

- **Percentiles:** Values that divide a dataset into 100 equal parts. The median is the 50th percentile.
- **Quartiles:** Values that divide a dataset into four equal parts. The first quartile (Q1) is the 25th percentile, the second quartile (Q2) is the median, and the third quartile (Q3) is the 75th percentile.

- **Skewness and Kurtosis:**

- **Skewness:** A measure of the asymmetry of the probability distribution of a real-valued random variable. Positive skewness indicates a right-skewed distribution, while negative skewness indicates a left-skewed distribution.
- **Kurtosis:** A measure of the "tailedness" or sharpness of the peak of a distribution. High kurtosis indicates heavy tails, while low kurtosis indicates light tails.

- **Correlation and Covariance:**
 - **Correlation Coefficient (Pearson):** A measure of the linear relationship between two variables. It ranges from -1 (perfect negative correlation) to 1 (perfect positive correlation).
 - **Covariance:** A measure of how much two variables change together. Positive values indicate a positive relationship, while negative values indicate a negative relationship.
- **Probability Distributions:**
 - **Normal Distribution:** A symmetric, bell-shaped distribution characterized by its mean and standard deviation. Many statistical methods assume data follows a normal distribution.
 - **Binomial Distribution:** Describes the number of successes in a fixed number of independent Bernoulli trials.
 - **Poisson Distribution:** Models the number of events occurring in a fixed interval of time or space.

- .

✓ **Hypothesis Testing:**

- ✓ **Null Hypothesis (H₀):** A statement assumed to be true, often representing no effect or no difference.
- ✓ **Alternative Hypothesis (H₁):** A statement that contradicts the null hypothesis, representing the effect or difference of interest.
- ✓ **Significance Level (α):** The threshold for deciding whether to reject the null hypothesis. Common choices include 0.05 and 0.01.

Confidence Intervals:

- ✓ **Confidence Interval:** A range of values within which the true parameter is likely to fall, based on sample data and a chosen level of confidence.

Regression Analysis:

- **Linear Regression:** A statistical method to model the relationship between a dependent variable and one or more independent variables.
- **Coefficient of Determination (R-squared):** Measures the proportion of the variance in the dependent variable explained by the independent variables.
- **Outliers:**
 - Outlier:** An observation that significantly differs from other observations in a dataset. Outliers can have a substantial impact on statistical analyses

Data Munging

- ✓ known as data wrangling or data preprocessing, refers to the process of cleaning, transforming, and organizing raw data into a format suitable for analysis.

Data Cleaning:

- ✓ **Handling Missing Values:** Identify and address missing data by either imputing values, removing rows or columns, or using advanced imputation techniques.
- ✓ **Dealing with Duplicates:** Identify and eliminate duplicate records to ensure data integrity.
- ✓ **Handling Outliers:** Identify and decide how to handle outliers, such as removing them or transforming their values.

Data Transformation:

- ✓ **Data Encoding:** Convert categorical variables into a numerical format, often using techniques like one-hot encoding or label encoding.
- ✓ **Data Scaling:** Normalize or standardize numerical features to bring them to a similar scale, preventing certain features from dominating others in machine learning models.
- ✓ **Feature Engineering:** Create new features or transform existing ones to capture relevant information and improve model performance.

Handling Text Data:

- ✓ **Text Cleaning:** Remove irrelevant characters, convert text to lowercase, and handle special characters.
- ✓ **Tokenization:** Break text into individual words or tokens for further analysis.
- ✓ **Stemming and Lemmatization:** Reduce words to their base or root form to standardize and simplify text data.

Dealing with Date and Time:

- ✓ **Date Parsing:** Convert date and time strings into a standard format for consistent analysis.
- ✓ **Extracting Features:** Extract relevant features from date and time data, such as day of the week, month, or year.

Handling Categorical Data:

- ✓ **One-Hot Encoding:** Convert categorical variables into binary vectors, creating a separate binary column for each category.
- ✓ **Label Encoding:** Assign numerical labels to categories, especially when there is an ordinal relationship among them.

Data Integration:

- ✓ **Combine Datasets:** Merge or concatenate multiple datasets to create a more comprehensive dataset for analysis.
- ✓ **Handling Inconsistent Data:** Address inconsistencies in data formats, units, or naming conventions across datasets.

Data Reduction:

- ✓ **Dimensionality Reduction:** Use techniques like Principal Component Analysis (PCA) or t-Distributed Stochastic Neighbor Embedding (t-SNE) to reduce the number of features while preserving important information.
- ✓ **Sampling:** If the dataset is large, consider using sampling techniques to create a representative subset for analysis.

Data Exploration:

- ✓ **Exploratory Data Analysis (EDA):** Visualize and analyze the distribution of data, identify patterns, and gain insights into potential relationships.
- ✓ **Data Validation and Sanity Checks:**
- ✓ **Check Data Integrity:** Ensure that data adheres to expected constraints and is consistent with domain knowledge.
- ✓ **Handle Inconsistent Formats:** Address inconsistencies in data formats and units.

Documentation:

- ✓ **Document Changes:** Keep a record of all transformations and cleaning steps applied to the data for reproducibility and transparency.

create a clean, consistent, and reliable dataset that reflects the underlying patterns and relationships

Databases for Data Science

SQL – Tool for Data Science

- Structured Query Language (SQL) is a language designed for **managing data in a relational database**.
- SQL has a variety of built-in functions that allow its users to **read, write, update, and delete data**.
- It is a popular language for data analysts for a few reasons:
 - It can **directly access a large amount of data** without copying it in other applications.
 - It can be used to deal with **data of almost any shape and massive size**.
 - Data analysis done in SQL is easy to audit and replicate as compared to an Excel sheet or CSV file. **Excel** is great with smaller datasets **but not useful for large-sized databases**.
 - **Joining tables, automating, and reusing code** are much easier in SQL than in Excel.
 - SQL is simple and easy to learn.

Basic Statistics with SQL: Mean

Mean Value

- The average of the dataset is calculated by dividing the total sum by the number of values (count) in the dataset.
- This operation can be performed in SQL by using built-in operation Avg.
- `SELECT Avg(ColumnName) as MEAN`

Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	USA

```
SELECT Avg(age) as Mean  
FROM Customers;
```

Mean

25.6

Basic Statistics with SQL: Mode

Mode Value

- The mode is the value that appears most frequently in a series of the given data.
- SQL does not provide any built-in function for this, so we need to write the following queries to calculate it.
- SELECT ColumnName

FROM TableName

GROUP BY ColumnName

ORDER BY COUNT(*) DESC

Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

```
SELECT age
FROM Customers
GROUP BY [age]
ORDER BY COUNT(*) DESC
LIMIT 1
```

Output

age
22

Basic Statistics with SQL: Median

Median value

- In a sorted list (ascending or descending) of numbers, the median value is the middle number and can be more descriptive of that dataset than the average.
- For an odd amount of numeric dataset, the median value is the number that is in the middle, with the same amount of numbers below and above.
- For an even amount of numbers in the list, the middle two numbers are added together, and this sum is divided by two to calculate the median value.
- SQL does not have a built-in function to calculate the median value of a column.

Basic Statistics with SQL: Median

Emp table with Ten
Employees Salary
Details

Salary	Name
1,000	Amit
2,000	Nisha
3,000	Yogesh
4,000	Puja
9,000	Ram
7,000	Husain
8,000	Risha
5,000	Anil
10,000	Kumar
6,000	Shiv

- Write the code that will return median value 5,500 for the given Emp table.
- In the above code, the internal subquery sorts salary and creates @rindex as an incremental index. The outer subquery will fetch the middle items in the array.
- If the dataset contains an even number of items, then the SELECT clause of the outer query calculates the average of those two values as the median value.

Data Munging with SQL

munging (or wrangling) is the phase of data transformation.

- It transforms data into various states so that it is simpler to work and understand the data.
- The transformation may lead to manually convert or merge or update the data manually in a certain format to generate data, which is ready for processing by the data analysis tools.
- Actually transform and map data from one format to another format to make it more valuable for a variety of analytics tools.
- Some frequently used data munging functions are:
 - UPPER()
 - LOWER()
 - TRIM()

Data Munging with SQL: UPPER()

- **UPPER()**
 - The UPPER() string function is used to convert the lower case to the upper case of the input text data.
 - **Input:** **SELECT UPPER('welcome to Data Science')**

UPPER('welcome to Data Science')

– WELCOME TO DATA SCIENCE

Data Munging with SQL: LOWER()

LOWER()

- The **LOWER()** string function is used to convert the upper case to the lower case of the input text data.

– **Input:**

SELECT LOWER('WELCOME TO DATA SCIENCE')

– **Output:**

```
LOWER('WELCOME TO DATA SCIENCE')
```

```
welcome to data science
```

Data Munging with SQL: TRIM()

- **TRIM()**
 - The TRIM() string function removes blanks from the leading and trailing position of the given input data.
 - **Input:** `SELECT TRIM('WELCOME TO DATA SCIENCE ')`
 - **Output:**

<code>TRIM(' WELCOME TO DATA SCIENCE')</code>
<code>WELCOME TO DATA SCIENCE</code>

Data Munging with SQL: TRIM()

- **LTRIM()**
 - The LTRIM() string function is used to remove the leading blank spaces from a given input string.

– **Input:**

```
SELECT LTRIM('      WELCOME TO  
DATA SCIENCE')
```

```
LTRIM(' WELCOME TO DATA SCIENCE')
```

```
WELCOME TO DATA SCIENCE
```

– **Output:**

Data Munging with SQL: RTRIM()


- **RTRIM ()**
 - The RTRIM() string function is used to remove the trailing blank spaces from a given input string.
 - **Input: SELECT RTRIM('WELCOME TO DATA SCIENCE ')**

– **Output:**

```
RTRIM('WELCOME TO DATA SCIENCE ')
```

```
WELCOME TO DATA SCIENCE
```

Data Munging with SQL: RIGHT()

- **RIGHT()**
 - The RIGHT() string function is used to return a given number of characters from the right side of the given input string.
 - **Input: SELECT RIGHT('WELCOME TO DATA SCIENCE', 7)**
 - **Output:** 

Data Munging with SQL: LEFT()

- **LEFT()**
 - The LEFT() string function is used to return a given number of characters from the left side of the given input string.
 - **Input: SELECT LEFT('WELCOME TO DATA SCIENCE', 7)**
 - **Output:**

WELCOME

Data Munging with SQL: REPLACE()

- **REPLACE()**

- The REPLACE() string function replaces all the occurrences of a source sub-string with a target substring of a given string.

- **Input:**

```
SELECT    REPLACE('WELCOME    TO    DATA
SCIENCE', 'DATA', 'INFORMATION')
```

- **Output:**

```
WELCOME TO INFORMATION SCIENCE
```

Filtering

Data filtering is one of the major steps involved in data science to remove redundant and useless data from the dataset and make the dataset more efficient and useful.

Need for Filtering

- During certain situations, we may require a specific part of the actual data for analysis.
- Sometimes, we may require reducing the actual retrieved data by removing redundant records as that may result in wrong analysis.
- Query performance can be greatly enhanced by applying it to refined data. Also, it can reduce strain on application.

Filtering

Syntax to filter data using WHERE

SELECT * FROM tablename;

The above query extracts all data from the table. An asterisk (*) in the above simple query indicates that “select all the data” in the table. In the above query, when we add WHERE clause with the condition after WHERE, it filters data in the table and returns only those records that satisfy the condition given after WHERE clause.

SELECT * FROM tablename

WHERE columnname = expected_value;

Filtering

Filter data using **WHERE**

Instead of the equal sign (=) operator in the condition statement of **WHERE**

clause, we can use the following operators too:

- > (greater than),
- < (less than),
- >= (greater than or equal to),
- <= (less than or equal to), and
- != (not equal to).

Filtering

Filter data using WHERE

Consider the following table

“workers” Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
John	11	30,000	301	Workshop
Jerry	15	35,000	305	Testing
Niya	38	45,000	308	HR
Alice	18	45,000	305	Testing
Tom	24	50,000	301	Workshop
Bobby	17	58,000	308	HR

Filtering

- Suppose we want to extract details of those employees who are working in “HR” department in the above workers table, then we can write the query as follows:

select * workers where DEPTNAME='HR';

The result of the above query is

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
Niya	38	45,000	308	HR
Bobby	17	58,000	308	HR

Filtering

- Suppose we want to extract details of those employees who are getting salary less than or equal to 47,000 in the above workers table, then we can write the query as follows:

select * from workers where SALARY<=47000;

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
John	11	30,000	301	Workshop
Jerry	15	35,000	305	Testing
Niya	38	45,000	308	HR
Alice	18	45,000	305	Testing

Filtering

- To fetch required data, sometimes, we may require to force two or more conditions. We can use AND, OR operators to achieve this. Only those records that satisfy all the conditions in the query will be retrieved when AND operator is used between two conditions. For example, to find workers in the HR department who have salary more than 47,000, we can write the query as follows

select * from workers where SALARY<=47000 AND DEPTNAME='HR';

The result of the above query is

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
Nirman	38	45000	308	HR

Filtering

If OR is used between two conditions, then all records that satisfy either condition will get retrieved along with records that satisfy both conditions. The following query will fetch the details of the workers who are working in the HR department or who have a salary less than 36,000

**select * from workers where SALARY<=36000 OR
DEPTNAME='HR';**

The result

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
John	11	30,000	301	Workshop
Jerry	15	35,000	305	Testing
Niya	38	45,000	308	HR
Bobby	17	58,000	308	HR

Filtering

To match a pattern in text data the LIKE clause can be used to specify a pattern matching condition.

Two wildcards, percent sign “%” and underscore “_”, are used to specify conditions.

The percent sign is used to represent any string of zero or more characters, and underscore represents a single number or character.

For example, to retrieve ENAME that ends with character “y” of workers table, we can write the query as follows:

select ENAME from workers where ENAME like '%y';

The result of

Result Table

ENAME
Jerry
Bobby

Filtering

Like clause

The following query retrieves records of salary, which has “5” at second position in workers table.

select SALARY from workers where SALARY like '_5%';

The result of the above query is

TABLE 3.14

Result Table

SALARY

35,000

45,000

45,000

Filtering

To filter records based on match of multiple values in a given dataset the SQL IN operator allows you to test if the given expression matches any value in the list of values.

If the records matched with any one of the values in the list, then it is returned as result. For example,

select * from workers where DEPTNAME IN('Testing', 'Workshop');

The above query returns the details of those workers whose DEPTNAME is “Workshop” or “Testing”

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
John	11	30,000	301	Workshop
Jerry	15	35,000	305	Testing
Alice	18	45,000	305	Testing
Tom	24	50,000	301	Workshop

Filtering

To exclude some values, NOT keyword in query. The following query returns those workers' details whose DEPTNAME is not "Workshop" or "Testing"

select * from workers where DEPTNAME NOT IN('Testing', 'Workshop');

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
Niya	38	45,000	308	HR
Bobby	17	58,000	308	HR

Filtering (Subquery)

- A subquery is a query within another query.
- A subquery (called a nested query or subselect) is a SELECT query embedded within the WHERE clause of another query.
- The data returned by the subquery (inner query) is used by the outer query in the same way literal values are used.
- A subquery is used to return data that will be used in the main or the outer query as a condition that the data must satisfy to be retrieved.
- Subqueries provide an easy way to handle the

Filtering (Subquery)

For example, in the following query, the inner query retrieves EID of workers who work in “HR” department or get salary \geq 40,000. The main query uses the result of the inner query and retrieves workers’ details whose EID matches with EIDs returned by the inner query

select * from workers where EID IN (select EID from workers where DEPTNAME='HR' OR SALARY \geq 40000).

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
Niya	38	45,000	308	HR
Alice	18	45,000	305	Testing
Tom	24	50,000	301	Workshop
Bobby	17	58,000	308	HR

Filtering (Subquery)

- To filter records based on match of a large range of values, use the keyword BETWEEN for this purpose.
- It allows you to specify a start value and an end value of required range. This clause is a shorthand representation for two conditions with >= and <= operators.

For example, to retrieve details of those workers having salary >= 30,000 and <= 45,000, write the query as follows

select * from workers where SALARY BETWEEN 30000 and 45000;

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
John	11	30,000	301	Workshop
Jerry	15	35,000	305	Testing
Niya	38	45,000	308	HR
Alice	18	45,000	305	Testing

Filtering

- To retrieve details of workers whose salary is not in the range of $\geq 30,000$ and $\leq 45,000$, write the query using NOT BETWEEN clause as follows
select * from workers where SALARY NOT BETWEEN 30000 and 45000;

Result Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
Tom	24	50,000	301	Workshop
Bobby	17	58,000	308	HR

Window Functions and Ordered Data

- In SQL window functions, the word “window” refers to a set of rows.
- It is similar to aggregate functions, but when we use the `GROUP BY` clause with aggregate functions, it groups the result set based on one or more columns.
- The aggregate function is performed on the rows as an entire group.
- SQL window functions generate a result with some attributes of an individual row together with the results of the window function .
- Window functions can be called with the `SELECT` statement or the `ORDER BY` clause, but cannot be called in the `WHERE` clause.
- The window function calculates on a set of rows and returns a value for each row from the given query.
- In the window function query syntax, the window is defined using the `OVER()` clause. In a single query, we can also include multiple window functions.

Window Functions

- The **OVER()** clause has the following sub clauses:
- **PARTITION BY** clause to define window partitions to form groups of rows on which window function will be applied.
- **ORDER BY** clause for logical sorting of rows within a partition.
- To demonstrate SQL window function, we will use the following “workers” table

“workers” Table

ENAME	EID	SALARY	DEPTID	DEPTNAME
John	11	30,000	301	Workshop
Jerry	15	35,000	305	Testing
Niya	38	45,000	308	HR
Alice	18	45,000	305	Testing
Tom	24	50,000	301	Workshop
Bobby	17	58,000	308	HR
Reyon	16	30,000	305	Testing
Bob	22	51,000	301	Workshop

Window Functions

RANK () Window Function

- The RANK() function returns the position of any row in the specified partition.
- The OVER and PARTITION BY functions are used to divide the result set into partitions according to specified criteria.
- Further, ORDER BY clause can be used to sort data in ascending or descending order based on some attribute.

Window Functions

To rank salaries within departments, we can write the query as follows:

```
SELECT RANK() OVER (PARTITION BY DEPTNAME ORDER BY  
SALARY DESC) AS DEPT_RANK, DEPTNAME, DEPTID,  
SALARY, ENAME FROM workers;
```

Result Table

DEPT_RANK	DEPTNAME	DEPTID	SALARY	ENAME
1	HR	308	58,000	Bobby
2	HR	308	45,000	Niya
1	Testing	305	45,000	Alice
2	Testing	305	35,000	Jerry
3	Testing	305	30,000	Reyon
1	Workshop	301	51,000	Bob
2	Workshop	301	50,000	Tom
3	Workshop	301	30,000	John

Window Functions

If we exclude PARTITION BY clause from the above query, then we will get the result as follows:

```
SELECT RANK() OVER (ORDER BY SALARY DESC)  
AS DEPT_RANK, DEPTNAME,DEPTID, SALARY, ENAME, EID  
FROM workers;
```

Result Table

DEPT_RANK	DEPTNAME	DEPTID	SALARY	ENAME
1	HR	308	58,000	Bobby
2	Workshop	301	51,000	Bob
3	Workshop	301	50,000	Tom
4	HR	308	45,000	Niya
4	Testing	305	45,000	Alice
6	Testing	305	35,000	Jerry
7	Workshop	301	30,000	John
7	Testing	305	30,000	Reyon

Window Functions

The RANK() function skips the rank 5 and rank 8 in the above result because two rows share the fourth rank and two records share the seventh rank. The RANK function skips the next $k-1$ ranks if there is a tie between k previous ranks.

Window Functions

PERCENT_RANK()

In SQL Server, the **PERCENT_RANK()** function calculates the SQL percentile rank of each row. This percentile ranking number may range from zero to one.

For each row, **PERCENT_RANK()** calculates the percentile rank using the following formula:

$$(\text{Rank} - 1) / (\text{total_number_rows} - 1)$$

In this formula, rank represents the rank of the row. total number_rows is the number of rows that are being evaluated. It always returns the rank of the first row as 0. It divides the rows into partitions by using the **PARTITION BY** clause, and the **ORDER BY** clause logically sorts (in ascending or descending order) rows for each partition. The percentile rank value is

Window Functions

PERCENT_RANK()

```
SELECT PERCENT_RANK() OVER (PARTITION BY DEPTNAME  
ORDER BY SALARY DESC) AS DEPT_RANK, DEPTNAME, DEPTID,  
SALARY, ENAME, EID FROM workers;
```

Result Table

DEPT_RANK	DEPTNAME	DEPTID	SALARY	ENAME	EID
0	HR	308	58,000	Bobby	17
1	HR	308	45,000	Niya	38
0	Testing	305	45,000	Alice	18
0.5	Testing	305	35,000	Jerry	15
1	Testing	305	30,000	Reyon	16
0	Workshop	301	51,000	Bob	22
0.5	Workshop	301	50,000	Tom	24
1	Workshop	301	30,000	John	11

Window Functions

PERCENT_RANK()

SELECT

**PERCENT_RANK() OVER (ORDER BY SALARY DESC) AS
DEPT_RANK, DEPTNAME, DEPTID, SALARY, ENAME, EID
FROM workers;**

Result Table

DEPT__PERCENT_ RANK	DEPTNAME	DEPTID	SALARY	ENAME	EID
0	HR	308	58,000	Bobby	17
0.14285714285714285	Workshop	301	51,000	Bob	22
0.2857142857142857	Workshop	301	50,000	Tom	24
0.42857142857142855	HR	308	45,000	Niya	38
0.42857142857142855	Testing	305	45,000	Alice	18
0.7142857142857143	Testing	305	35,000	Jerry	15
0.8571428571428571	Workshop	301	30,000	John	11
0.8571428571428571	Testing	305	30,000	Reyon	16

Preparing Data for Analytics Tool.

- One of the primary steps performed for data science is the cleaning of the data set .
- The valuable information or patterns extract from dataset are just in the same class as the data itself, so maximum of the time spent by a data scientist or analyst includes preparing datasets for use in analysis.
- SQL can help to speed up this step. Various SQL queries can be used to clean, update, and filter data, by eliminating redundant and unwanted records.
- This can be done with the different SQL clauses like CASE WHEN, COALESCE (**returns the first non-null value in a list.**), NULLIF, LEAST/GREATEST, Casting, and DISTINCT.

Preparing Data for Analytics Tool.

"sales" Table

sale_no	product_id	quantity	price	customer_name
5,001	3	4	21,000	John
5,002	11	NULL	17,000	Anna
5,003	94	10	105,000	Tom
5,004	86	8	27,000	Nora
5,005	88	18	8,000	Tom

Preparing Data for Analytics

CASE WHEN Tool

The **CASE** statement goes through various conditions specified with **WHEN** clause and returns a value when the first condition is met. It works like nested **IF-THEN-ELSE** statement. Once a condition is true, it will return the value specified after **THEN**. Value in the **ELSE** clause is returned, if no conditions are true. It returns **NULL** when no conditions are true, and no **ELSE** part is specified in the query.

Suppose we fetch all data of the above sales table and want to add an extra column that labels as summary which categorizes sales into More, Less, and Avg, this table can be created using a **CASE** statement as follows:

```
SELECT *,  
CASE  
WHEN quantity >= 10 THEN 'More' WHEN quantity >= 6 THEN  
'Avg' ELSE 'Less'
```

Preparing Data for Analytics Tool.

Result Table

sale_no	product id	quantity	price	customer_name	summary
5,001	3	4	21,000	John	Less
5,002	11	NULL	17,000	Anna	Less
5,003	94	10	105,000	Tom	More
5,004	86	8	27,000	Nora	Avg
5,005	88	10	8,000	Tom	More

Preparing Data for Analytics

COALESCE Tool.

Some records of database may consist of NULL values, but while applying statistics to these datasets, you may need to replace these NULL values with some other data. This can be done effectively by the COALESCE function. The first parameter to this function is a column that may consist of NULL, and the second represents value that replaces NULL. It replaces all NULL values specified in column by the second default value given in the function.

The following example replaces NULL by -1 in the quantity column.

```
SELECT  
customer_name ,product_id,  
COALESCE(quantity, -1) AS quantity FROM
```

Preparing Data for Analytics

Tool
TABLE 3.44

Result Table

customer_name	product_id	quantity
John	3	4
Anna	11	-1
Tom	94	10
Nora	86	8
Tom	88	10

Preparing Data for Analytics Tool.

NULLIF

NULLIF function takes two parameters and will return NULL if the first parameter value equals the second value else returns the first parameter.

As an example, imagine that we want to replace product_id value 11 by NULL.

SELECT sale_no, customer_name, NULLIF(product_id, 11) AS product_id FROM sales;

Result Table

sale_no	customer_name	product id
5,001	John	3
5,002	Anna	NULL
5,003	Tom	94
5,004	Nora	86

Preparing Data for Analytics Tool.

LEAST/GREATEST

The **LEAST** and **GREATEST** are frequently used functions for data cleaning.

These functions return the least and greatest values from the given set of elements, respectively.

These functions are useful to replace value in list, especially if it is too high or low.

For example, minimum price needs to be 10,000 in the above table. This can be done by the following query. Price 8,000 is replaced by value 10,000 in the last row, as 8,000 is less than 10,000, and it replaces it by max value among these two.

Preparing Data for Analytics Tool.

SELECT sale_no, product_id, quantity, GREATEST(10000, price) as price FROM sales;

Result Table

sale_no	product_id	quantity	price
5,001	3	4	21,000
5,002	11	NULL	17,000
5,003	94	10	105,000
5,004	86	8	27,000
5,005	88	10	10,000

Preparing Data for Analytics Tool.

DISTINCT

The DISTINCT keyword returns only distinct values in the specified column value sets.

For example, to extract all the unique names in the sales table, you could write the following query:

SELECT

DISTINCT customer_name FROM sales;

DISTINCT clause can also be applied to multiple columns to get the distinct combinations of the specified column. The above query gives the following result: It removed duplicate names from the customer name column

Result Table

customer_name
John
Anna
Tom
Nora

Advanced NoSQL for Data Science

- NoSQL is a database design that can accommodate various data models, including key-value, document, columnar, and graph formats.
- NoSQL, which means “not only SQL”, is an alternative to relational databases in which data is stored in tables and has a fixed data schema. NoSQL databases are very useful for working with large distributed data. The NoSQL databases are built in the early 2000s to deal with large-scale database clustering in web and cloud applications.
- NoSQL has a **flexible schema**, unlike the traditional relational database model.

Advanced NoSQL for Data Science

- NoSQL databases are found to be very useful for handling really big data tasks because it follows the Basically Available, Soft State, Eventual Consistency (**BASE**) approach instead of Atomicity, Consistency, Isolation, and Durability commonly known as **ACID properties**.
- Two major drawbacks of SQL are **rigidity** when adding columns and attributes to tables and **slow performance** when many tables need to be joined and when tables store a large amount of data. NoSQL databases tried to overcome these two biggest drawbacks of relational databases.

Why NoSQL

NoSQL supports unstructured data or semi-structured data. In many applications, an attribute usually needs to be added on the fly, for specific rows, but not every row, and may be of different types than attributes in the rows. Now let us explore some NoSQL features to understand why you should choose NoSQL databases for data science.

Features:

- It is not using the relational model to store data.
- NoSQL running well on clusters.
- It is mostly open-source.
- NoSQL is capable to handle a large amount of social media data.
- NoSQL is schema-less.

Categories of NoSQL

- Document Databases
- Key-value Databases
- Graph Databases.

Document Databases

- Document-based NoSQL databases store the data in the JSON object format.
- Each document has key-value pairs like structures.
- The document-based NoSQL databases are simple for engineers as they map items as a JSON object.
- JSON is a very common data format truly adaptable by web developers and permits us to change the structure whenever required.
- Some examples of document-based NoSQL databases are CouchDB, MongoDB, OrientDB, and BaseX.

Document Databases

JSON document format:

```
{
  "_id": 1,
  "name" : { "first" : "John", "last" : "Backus" },
  "contribs" : [ "Fortran", "ALGOL", "Form", "FP" ], "awards" : [
    {
      "award" : "Dowell Award", "year" : 1988,
      "by" : "Computer Society"
    },
    {
      "award" : "First Prize", "year" : 1993,
      "by" : "National Academy of Engineering"
    }
  ]
}
```

✓ **A document database (also known as a document-oriented database or a document store) is a database that stores information in documents.**

✓ Document is a record in a document database. A document typically stores information about one object and any of its related metadata.

✓ Documents store data in field-value pairs.

✓ The values can be a variety of types and structures, including strings, numbers, dates, arrays, or objects.

✓ Documents can be stored in formats like JSON, [BSON](#), and XML

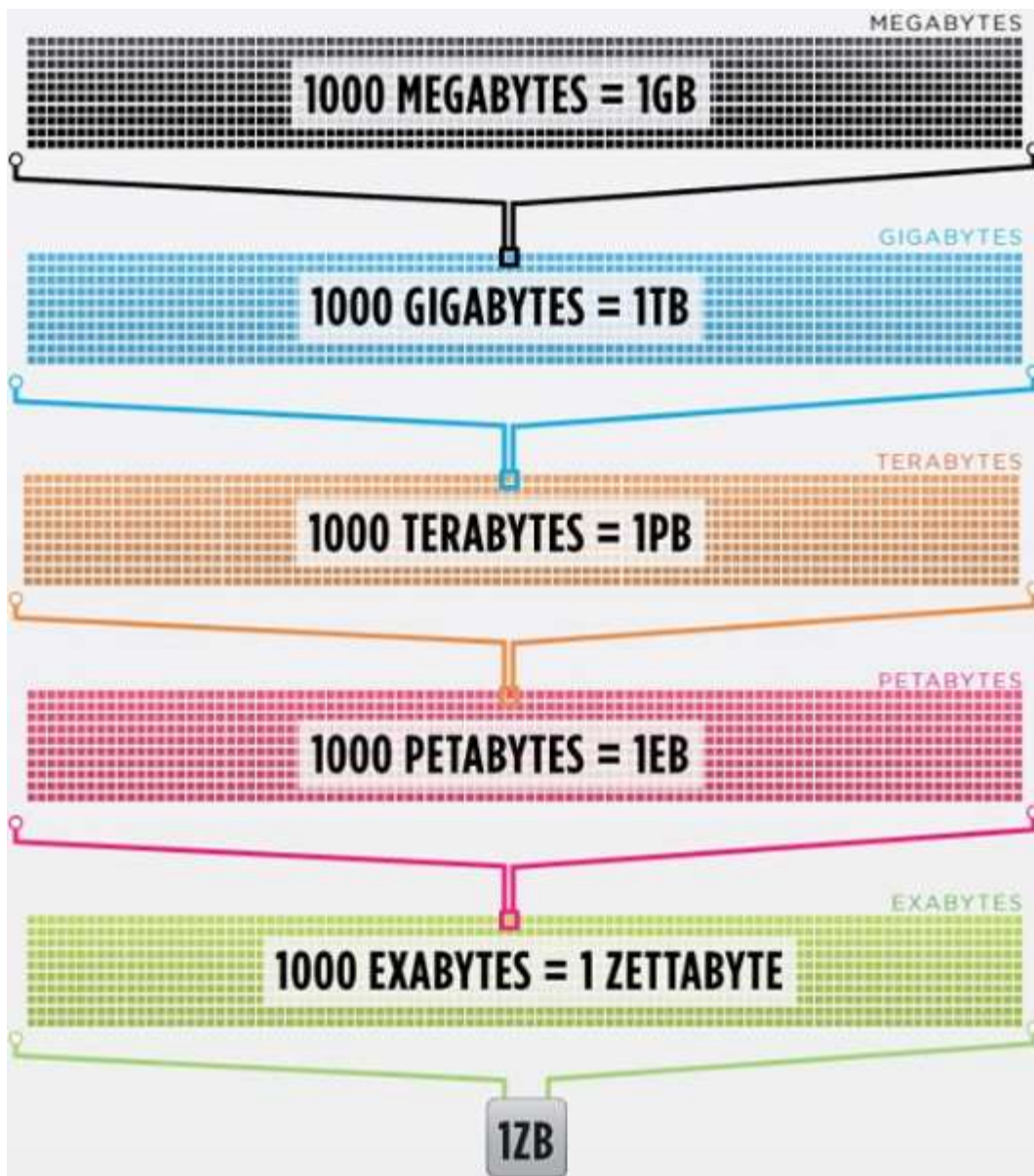
CRUD operations

have an API or query language that allows developers to execute the CRUD (create, read, update, and delete) operations.

- ✓ **Create:** Documents can be created in the database. Each document has a unique identifier.
- ✓ **Read:** Documents can be read from the database. The API or query language allows developers to query for documents using their unique identifiers or field values. Indexes can be added to the database in order to increase read performance.
- ✓ **Update:** Existing documents can be updated — either in whole or in part.
- ✓ **Delete:** Documents can be deleted from the database

- Trends in Data
- What is a Graph?
- What is a Graph Database?
- What is Neo4j?

Trends in Data



Data is getting bigger:

“Every 2 days we create as much information as we did up to 2023”

Data is more connected:

- Text (content)
- HyperText (added pointers)
- RSS (joined those pointers)
- Blogs (added pingbacks)
- Tagging (grouped related data)
- RDF (described connected data)
- GGG (content + pointers + relationships + descriptions)

✓ A graph database is a specialized, single-purpose platform used to create and manipulate data of an associative and contextual nature.

✓ The graph itself contains nodes, edges, and properties that come together to allow users to represent and store data in a way that relational databases aren't equipped to do.

✓ The main concept of a graph database system is a Relationship.

✓ Relationships allow data within the system to be linked together directly.

✓ Querying relationships in a graph database is fast since they're stored in a way that doesn't change.

✓ You may also visualize them, which makes them great for deriving insights for heavily interconnected data

Graph?

- An abstract representation of a set of objects where some pairs are connected by links.



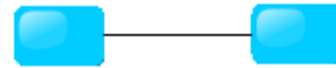
Object (Vertex, Node)



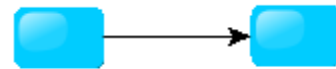
Link (Edge, Arc, Relationship)

Different Kinds of Graphs

- Undirected Graph



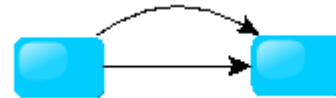
- Directed Graph



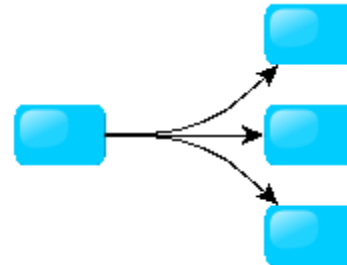
- Pseudo Graph



- Multi Graph

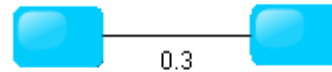


- Hyper Graph

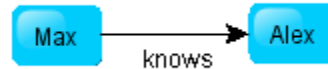


More Kinds of Graphs

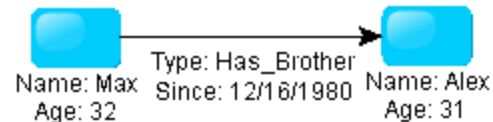
- Weighted Graph



- Labeled Graph



- Property Graph



Graph Database?

- A database with an explicit graph structure
- Each node knows its adjacent nodes
- As the number of nodes increases, the cost of a local step (or hop) remains the same
- Plus an Index for lookups

Examples

Tiger Graph

- Tiger Graph provides a user-friendly query language called GSQL - designed for graph database queries. Easier for developers and analysts to interact with and extract valuable information
 - D Graph – Ratel
 - Stardog
 - Neo 4j

✓ Actor

✓ Relational tie

✓ Dyad

✓ Triad

✓ Subgroup

✓ Group

✓ Relation

✓ Network

ACTOR

- ✓ Understanding the linkage among social entities and implications.
- ✓ Discrete individual, Corporate or collective social units.
- ✓ Example : People in a group, Departments within university

One Mode Networks:

- ✓ Focus on actors collections that are of same type (People in a group)

Relational Tie

- ✓ Actors are linked to one another by social ties
- ✓ Establishing links between pair of actors
- ✓ Evaluation of one person by another
- ✓ Transfer of material resources (Business Transactions, Lending or borrowing)
- ✓ Association of Affiliation (Jointly attending a social event, Belong to same social club)
- ✓ Behavioural Interactions (Talking together, Sending messages)
- ✓ Movement between places or status (Migration, Social or physical mobility)
- ✓ Formal Relationship, Biological Relationship(kinship)

Dyad

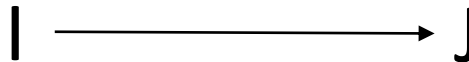
- ✓ Establishes relationship between two actors
- ✓ Consists of pair of actors and possible ties between them
- ✓ Analysis and focuses on pair wise Relation
- ✓ Ties has reciprocated or not
- ✓ Specific types of multiple relationship tend to occur together

Triad

- ✓ Relationship among larger subset of actors
- ✓ Many social network methods and models focus on **TRIAD**

K (subset of three Actors and possible ties among them)

- ✓ Balanced theory has informed and motivated many triadic Analysis



Sub Group, Group

- ✓ Dyads pair of actors Triads Triples of Actors
- ✓ Subgroup of actors as any subset of actors and ties among them

Group

- ✓ Collection of all actors on which ties are to be measured
- ✓ Consists of finite set of actors who for conceptual, theoretical or empirical reasons are treated as finite set of individuals on which network measurements are made.
- ✓ Specification of network boundaries, Sampling, Definition of group
–Problematic Issue

Relation

- Collection of ties of specific kind among members of a group.
- Set of friendship among pair of students in a classroom
- Set of formal diplomatic ties maintained by pair of nations in the world

Social network

- Consists of finite set or set of actors and the relation defined on them

Three type of notations

- ✓ Graph Theoretic
- ✓ Algebraic
- ✓ Sociometric

Things you can do in a Social Network

- ✓ Communicating with existing networks, making and developing friendships/contacts
- ✓ Represent themselves online, create and develop an online presence
- ✓ Viewing content/finding information
- ✓ Creating and customizing profiles
- ✓ Authoring and uploading content
- ✓ Adding and sharing content
- ✓ Posting messages – public & private
- ✓ Collaborating with other people

Definition of Social Networks

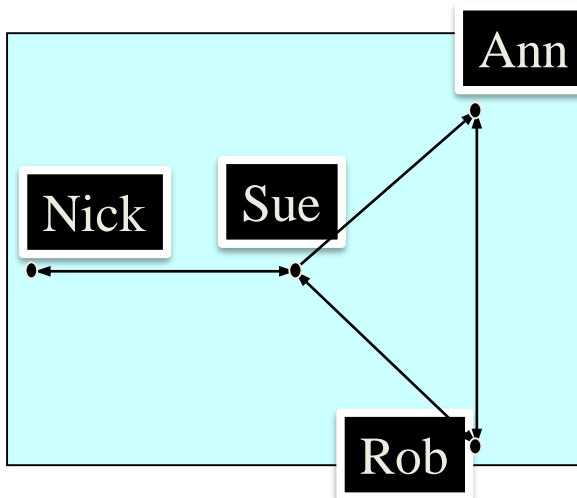
“A social network is a set of actors that may have relationships with one another. Networks can have few or many actors (nodes), and one or more kinds of relations (edges) between pairs of actors.”

Representation of Social Networks

✓ Matrices

	Ann	Rob	Sue	Nick
Ann	---	1	0	0
Rob	1	---	1	0
Sue	1	1	---	1
Nick	0	0	1	---

✓ Graphs



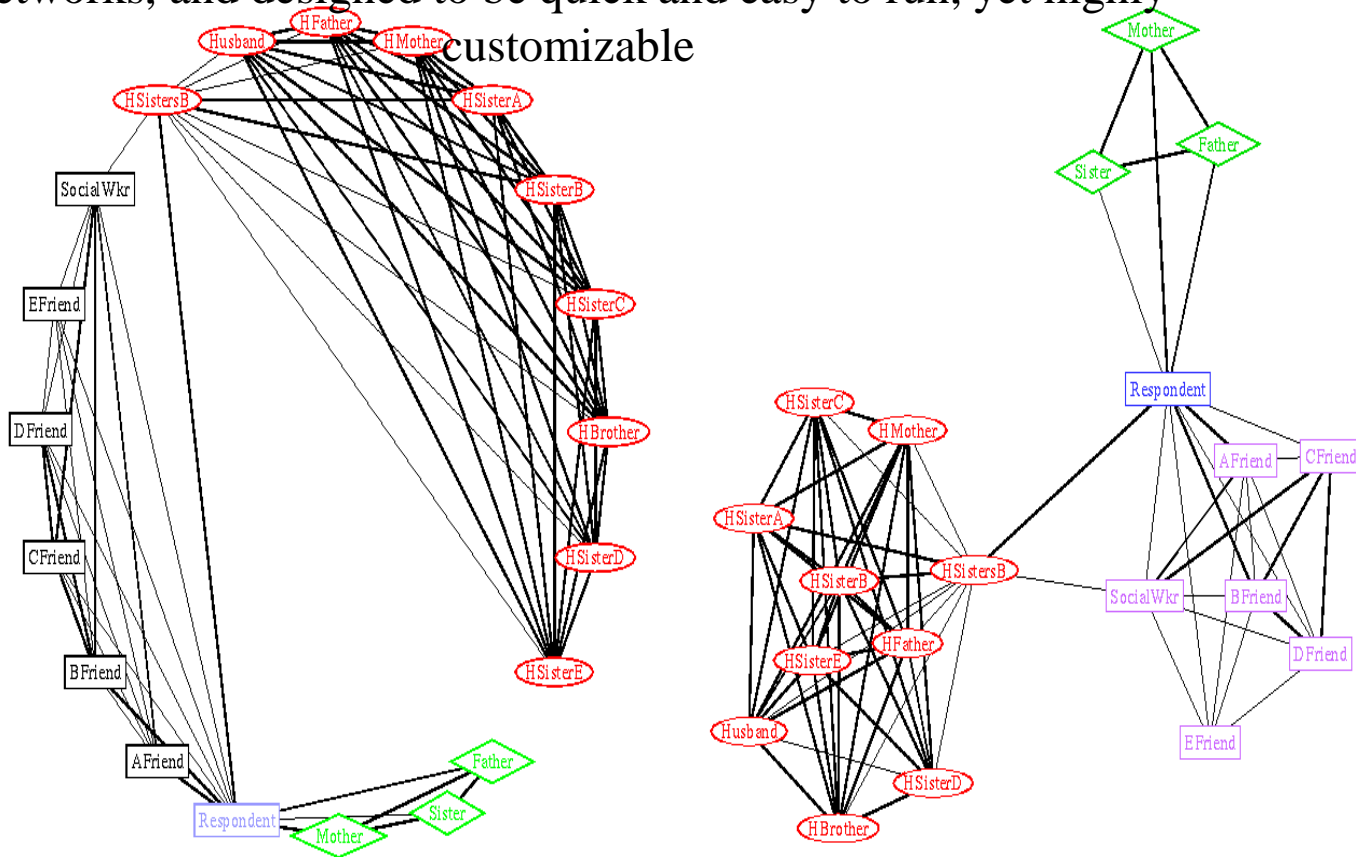
Graphs - Sociograms

- ✓ Labeled circles represent actors
- ✓ Line segments represent ties
- ✓ Graph may represent one or more types of relations
- ✓ Each tie can be directed or show co-occurrence
- ✓ Arrows represent directed ties

Visualization Software:

Krackplot

KrackPlot is a network visualization tool intended for social networks, and designed to be quick and easy to run, yet highly customizable



Connections

- ✓ Size
 - ✓ Number of nodes
- ✓ Density
 - ✓ Number of ties that are present vs the amount of ties that could be present
- ✓ Out-degree
 - ✓ Sum of connections from an actor to others
- ✓ In-degree
 - ✓ Sum of connections to an actor
- ✓ Diameter
 - ✓ Maximum greatest least distance between any actor and another

Some Measures of Distance

- ✓ Walk (path)

- ✓ A sequence of actors and relations that begins and ends with actors

- ✓ Geodesic distance (shortest path)

- ✓ The number of actors in the shortest possible walk from one actor to another

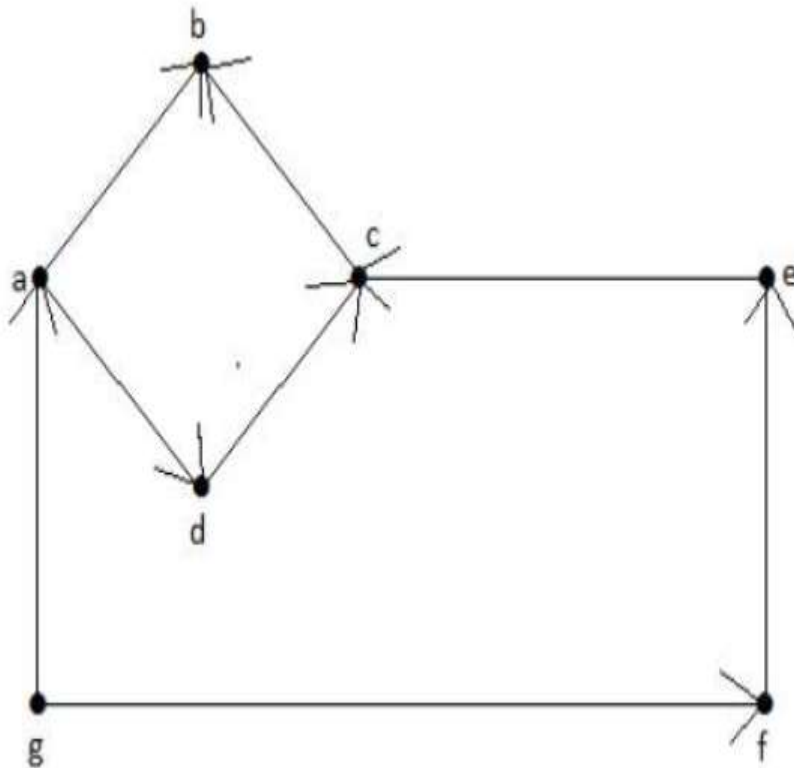
- ✓ Maximum flow

- ✓ The amount of different actors in the neighborhood of a source that lead to pathways to a target

Some Measures of Power

- ✓ Degree (indegree, outdegree)
 - ✓ Sum of connections from or to an actor
- ✓ Closeness centrality
 - ✓ Distance of one actor to all others in the network
- ✓ Betweenness centrality
 - ✓ Number that represents how frequently an actor is between other actors' geodesic paths

Calculate the in degree and out degree for the diagram given below



Vertex	In degree	Out Degree
--------	-----------	------------

Cliques and Social Roles

✓ Cliques

✓ Sub-set of actors

- ✓ More closely tied to each other than to actors who are not part of the sub-set

✓ Social roles

- ✓ Defined by regularities in the patterns of relations among actors

SNA applications

- ✓ Many new unexpected applications plus many of the old ones
 - ✓ Marketing
 - ✓ Advertising
- ✓ Economic models and trends
- ✓ Political issues
 - ✓ Organization
- ✓ Services to social network actors
 - ✓ Travel; guides
 - ✓ Jobs
 - ✓ Advice
- ✓ Human capital analysis and predictions
- ✓ Medical
- ✓ Epidemiology
- ✓ Defense (terrorist networks)

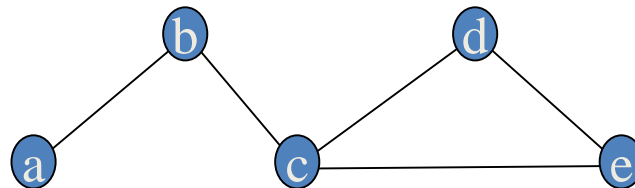
Foundations

- ✓ The unit of interest in a network are the combined sets of actors and their relations.
- ✓ We represent *actors* with points and *relations* with lines.

Actors are referred to variously as: Nodes, vertices, actors or points

Relations are referred to variously as: Edges, Arcs, Lines, Ties

Example:



Foundations

Data

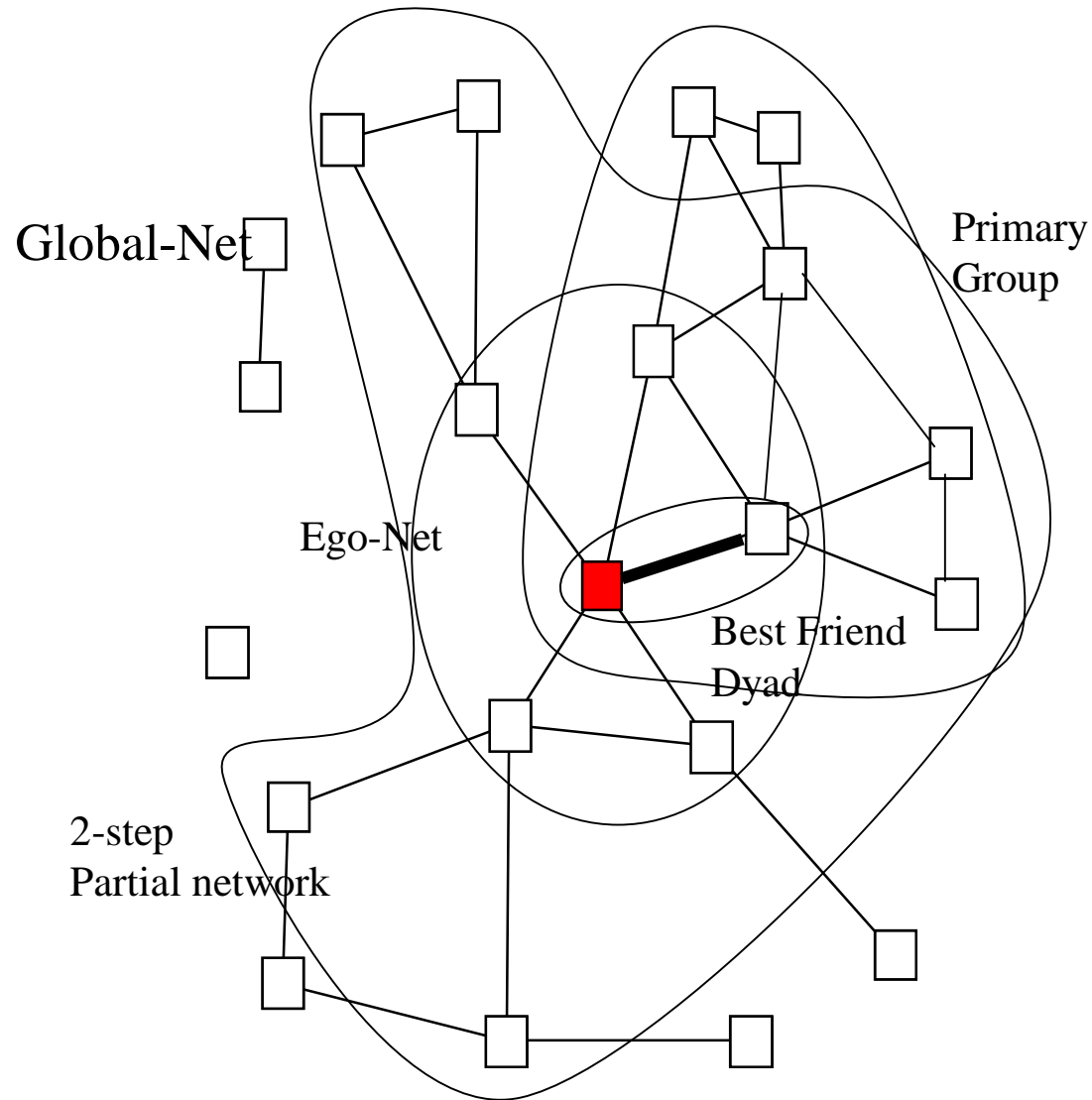
Social Network data consists of two linked classes of data:

- a) **Nodes:** Information on the individuals (actors, nodes, points, vertices)
 - Network nodes are most often people, but can be any other unit capable of being linked to another (schools, countries, organizations, personalities, etc.)
 - The information about nodes is what we usually collect in standard social science research: demographics, attitudes, behaviors, etc.
 - Often includes dynamic information about when the node is active

- b) **Edges:** Information on the relations among individuals (lines, edges, arcs)
 - Records a connection between the nodes in the network
 - Can be valued, directed (arcs), binary or undirected (edges)
 - One-mode (direct ties between actors) or two-mode

Foundations

Data and social science



Foundations

Data

We can examine networks across multiple levels:

1) Ego-network

- Have data on a respondent (ego) and the people they are connected to (alters). Example: terrorist networks
- May include estimates of connections among alters

2) Partial network

- Ego networks plus some amount of tracing to reach contacts of contacts
- Something less than full account of connections among all pairs of actors in the relevant population
- Example: CDC Contact tracing data

- ✓ [Networks with ground-truth communities](#) : ground-truth network communities in social and information networks
- ✓ [Social networks](#) : online social networks, edges represent interactions between people
- ✓ [Communication networks](#) : email communication networks with edges representing communication
- ✓ [Citation networks](#) : nodes represent papers, edges represent citations
- ✓ [Collaboration networks](#) : nodes represent scientists, edges represent collaborations (co-authoring a paper)
- ✓ [Web graphs](#) : nodes represent webpages and edges are hyperlinks
- ✓ [Amazon networks](#) : nodes represent products and edges link commonly co-purchased products
- ✓ [Internet networks](#) : nodes represent computers and edges communication
- ✓ [Road networks](#) : nodes represent intersections and edges roads connecting the intersections
- ✓ [Autonomous systems](#) : graphs of the internet
- ✓ [Signed networks](#) : networks with positive and negative edges (friend/foe, trust/distrust)
- ✓ [Location-based online social networks](#) : Social networks with geographic check-ins

Wide-Column Databases for Data Science

- Similar to any relational database, this wide-column database stores the data in **records**, but it can also store very large numbers of **dynamic columns**.
- It groups the dynamically added columns into column families. Instead of having multiple tables like relational databases, we have multiple column families in wide-column databases.

Wide-Column Databases for Data

name
value

Column

super column name		
name	...	name
value		value

Super
Column

row key	name	...	name
	value		value

Column
Family

row key	super column name			...	super column name		
	name	...	name		name	...	name
	value		value		value		value

Super
Column
Family