PLATFORM AND ARCHITECTURE FOR DATA SCIENCE(P&A):

ROLES AND THEIR RESPONSIBILITIES:

1. Business Analyst

- **Responsibilities**: Requirement gathering, stakeholder communication, process improvement.
- **Skills Required**: Communication, documentation, data visualization, domain knowledge.

2. BI (Business Intelligence) Analyst

- **Responsibilities**: Reporting, data visualization, dashboard creation.
- **Skills Required**: SQL, BI tools (e.g., Power BI, Tableau), data warehousing.

3. Data Analyst

- **Responsibilities**: Data cleaning, statistical analysis, reporting.
- **Skills Required**: Excel, SQL, Python/R, data visualization.

4. ML (Machine Learning) Engineer

- **Responsibilities**: Model development, model deployment, data preprocessing.
- **Skills Required**: Python, ML libraries (e.g., TensorFlow, scikit-learn), cloud services.

5. ML/AI Developer

- **Responsibilities**: Implementing AI models, integrating ML algorithms.
- **Skills Required**: Programming (Python, Java), ML frameworks, deep learning.

6. Data Engineer

- **Responsibilities**: Data pipeline development, ETL (Extract, Transform, Load), data integration.
- **Skills Required**: SQL, Python, ETL tools, cloud platforms.

7. Data Architect

- **Responsibilities**: Data modeling, database design, architecture development.
- **Skills Required**: Data modeling tools, SQL, big data technologies, cloud infrastructure.

8. Analytical Engineer

- **Responsibilities**: Data pipeline design, analysis optimization, metric development.
- **Skills Required**: SQL, Python, data modeling, analytics tools.

9. Platform Engineer

- **Responsibilities**: Infrastructure management, platform optimization, deployment.
- **Skills Required**: Cloud computing, Kubernetes, CI/CD, DevOps.

10. Research Scientist

- **Responsibilities**: Research, experimentation, algorithm development.
- **Skills Required**: Advanced statistics, ML techniques, research methodologies, academic writing.

11. Product Manager

- **Responsibilities**: Product strategy, roadmap planning, cross-functional coordination.
- **Skills Required**: Communication, project management, market research, analytical thinking.

Data Engineering

- **Definition**:
  The process of designing and building systems to collect, store, and analyze data, ensuring it's clean, accessible, and usable for analytics or machine learning.
- **Real-world Example**:
  Think of data engineers as people who build roads for data. They create pipelines to move data from one place to another, like constructing roads to transport goods from a factory to stores.

Streaming

- **Definition**:
  Real-time data processing where data is analyzed as soon as it's created or received.
- **Real-world Example**:
  Imagine a security camera recording live. The footage is sent directly to your phone, where you can see it happening in real time. That's streaming!

Batch Processing

- **Definition**:
  Collecting and processing data in chunks at specific intervals (e.g., hourly, daily).
- **Real-world Example**:
  Picture a bakery that makes bread only once a day, at the end of the day. It collects all orders throughout the day and then bakes them in a batch. That's batch processing.

Data Warehouse:

- **Definition**:
  A system for storing large volumes of structured data, optimized for analysis and reporting.
- **Real-world Example**:
  Imagine a library storing books (data) neatly categorized and organized by genres (tables) so that you can easily find what you need.

Data Lake:

- **Definition**:
  A storage system for holding raw and unstructured data (like documents, images, videos) until needed for processing.
- **Real-world Example**:
  Think of a data lake as a huge, messy storage room where you throw everything (books, old toys, files). You can organize or analyze it later, depending on what you need.

Data Strategy and Transformation Plan:

1. Understand the Business System

- Before you start working with data, it's crucial to **understand the business environment** and identify potential **challenges** or **gaps**.
- **Analyze the business needs** thoroughly to see how data can support those needs effectively.

Example:

Imagine you join a company that sells sports equipment online. Start by identifying what their customers want (e.g., faster deliveries, product recommendations) and where the company might be facing issues (e.g., tracking inventory).

---

2. Identify Data Sources and Frequency

- Figure out **what data is needed** and **how often** it should be updated.
- Determine the types of data to collect (e.g., customer purchase history, web traffic data).
- Choose the appropriate storage solutions, such as **databases, data warehouses**, or **data lakes**.

Example:

For the sports equipment company, you need data on product sales, customer preferences, and inventory. This data should be updated **daily** for inventory and **weekly** for customer preferences.

---

3. Optimization for Media Platforms

- If the business involves media streaming (e.g., Netflix), understand the need for **real-time data processing**.
- Choose methods that help optimize the performance for better user experiences.

Example:

If working for Netflix, use streaming data to show users the most popular movies in real-time.

---

4. Choosing the Right Data Processing Strategy

- Decide whether the data should be processed in **batches** (e.g., daily reports), **streamed** (e.g., real-time updates), or handled in **real-time** (e.g., instant notifications).

Example:

For customer feedback, real-time processing can immediately show new comments, while weekly sales trends can be handled using batch processing.

---

5. Data Storage Systems

- Use **databases** for structured data (e.g., customer info), **data lakes** for unstructured data (e.g., videos), and **data warehouses** for comprehensive reporting.

- Choose systems that can **store and retrieve data quickly**.

Example:

Store customer names and emails in a database, while large product images go into a data lake for easier access.

---

6. Business Transformation through Centralization

- Centralize your data so it can be easily accessed for tasks like **machine learning** and **detailed analysis**.
- This helps improve decision-making and streamline business operations.

Example:

Instead of storing sales data separately in each store, centralize it in a data warehouse to get a complete view of total sales.

---

7. Process Automation

- Identify business processes that are repetitive and prone to errors (e.g., manual data entry).
- Use automation tools to make these processes faster and error-free.

Example:

Set up a system that automatically updates inventory levels based on sales without human intervention.

---

8. Practical Implementation

- Make sure all steps are aligned with the business goals.
- Use tools like **Spark** and **PySpark** for managing large data sets.
- Set up schedulers to automate data updates or syncs.

Example:

Use PySpark to analyze customer reviews and set a scheduler to update this analysis every night.

3-Year Data Roadmap

- **Phase 1 (Year 1):** Start by setting up the **data foundation** – gather data sources, build storage, and understand processing needs.
- **Phase 2 (Year 2):** Scale up operations to handle more complex data and accommodate business growth.
- **Phase 3 (Year 3):** Lead with data by making data-driven decisions and establish the company as a leader in data strategy.

Example:

For the sports equipment company, Year 1 is focused on setting up a database for orders. Year 2 scales to include more data like customer feedback. Year 3 uses this combined data for predicting trends and improving sales strategies.

D&A Rebranding Strategy

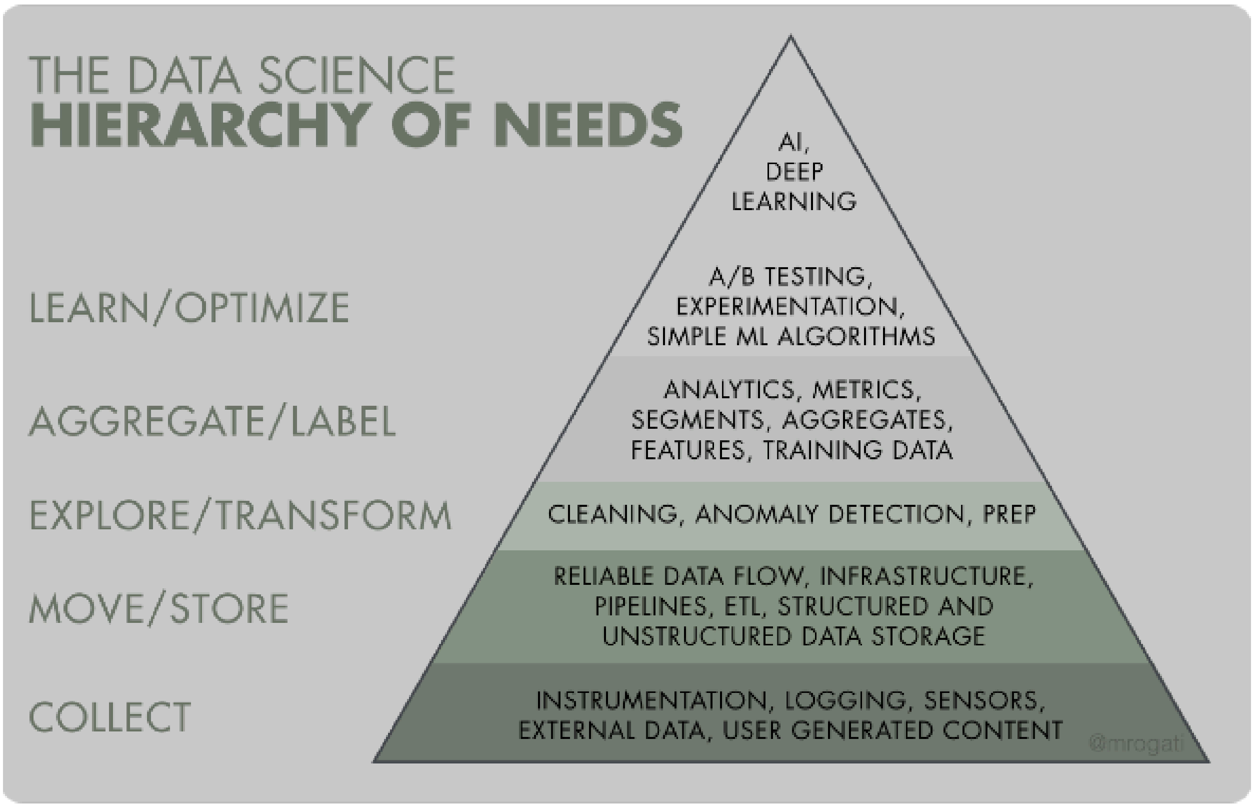
1. **Gather Information About the Client**
   - Understand who your client is and how they want to work with you.
   - **Example:** If the client is a tech startup, learn about their current projects and business goals to see how you can support them.
2. **Share Your Previous Work**
   - Let them know about the projects you've completed before and what you achieved.
   - **Example:** If you helped a company increase their sales through a digital marketing campaign, mention this to show your expertise.
3. **Ask Relevant Questions**
   - Ask specific questions to understand their project better.
   - **Example:** “What is your main objective with this project—do you want more customers, better brand visibility, or something else?”
4. **Highlight Your Past Solutions**
   - Show them how you have previously solved similar problems for other clients.
   - **Example:** “We faced a similar challenge with a client in retail. By revamping their online store, we boosted their sales by 30%.”

Our Approach: Swift and Sophisticated Solutions

- **Swift Solutions:**
  - We solve your problems quickly and effectively.
  - **Example:** If your website is facing downtime, we’ll find the issue and fix it swiftly to minimize the impact.

- **Sophisticated Solutions:**
    - The more complex your problem is, the more eager we are to find a unique and advanced solution for you.
    - **Example:** If you have data scattered across multiple systems, we can design an integrated database that centralizes all your data for easy access.
- **Why Us?**
    - We believe that every problem has a solution. Our experienced and analytical team is ready to offer tailored solutions for your business.

THE DATA SCIENCE HIERARCHY OF NEEDS:


THE DATA SCIENCE
HIERARCHY OF NEEDS
AI, DEEP LEARNING
LEARN/OPTIMIZE
A/B TESTING, EXPERIMENTATION, SIMPLE ML ALGORITHMS
AGGREGATE/LABEL
ANALYTICS, METRICS, SEGMENTS, AGGREGATES, FEATURES, TRAINING DATA
EXPLORE/TRANSFORM
CLEANING, ANOMALY DETECTION, PREP
MOVE/STORE
RELIABLE DATA FLOW, INFRASTRUCTURE, PIPELINES, ETL, STRUCTURED AND UNSTRUCTURED DATA STORAGE
COLLECT
INSTRUMENTATION, LOGGING, SENSORS, EXTERNAL DATA, USER GENERATED CONTENT
@mrogati

1. COLLECT:

- **Definition**: This is the foundation of the pyramid, where data is gathered from various sources.
- **Explanation**: It includes collecting data using tools, sensors, or systems that record information. It could be user-generated content (like social media posts), data from external sources, or logs from software.
- **Example**: Imagine a weather app collecting temperature and humidity data from various sensors placed in different cities.

2. MOVE/STORE:

- **Definition**: This stage involves transferring and storing the collected data.
- **Explanation**: Data should be moved to a storage system where it can be easily accessed and managed. The goal is to have a reliable and efficient way of storing all the data.
- **Example**: Transferring weather data from sensors to a central database for further analysis and keeping it organized.

3. EXPLORE/TRANSFORM:

- **Definition**: Preparing and cleaning the data for analysis.
- **Explanation**: In this step, the data is cleaned, formatted, and processed to ensure it is accurate and ready for use. Any errors, missing values, or anomalies are identified and resolved.
- **Example**: Removing incorrect temperature readings or filling in missing values in the weather data.

4. AGGREGATE/LABEL:

- **Definition**: Grouping and labeling the data.
- **Explanation**: Data is aggregated to create useful summaries or labeled to identify different types of information. This helps in organizing the data better for analysis.
- **Example**: Grouping the weather data by regions and labeling each region as "coastal," "urban," or "mountainous."

5. LEARN/OPTIMIZE:

- **Definition**: Applying advanced algorithms and optimization techniques.
- **Explanation**: Using machine learning or deep learning models to learn from the data and make predictions or optimizations. This stage involves using sophisticated techniques to gain insights or create intelligent systems.
- **Example**: Building a machine learning model that predicts weather patterns, such as storms or heatwaves, based on historical data.

Each stage builds upon the previous one, and you need a strong foundation (like collecting good quality data) before moving up to more complex tasks (like using AI or deep learning).

DATA ENGINEERING BALANCING ACT:

Explanation of Terms:

1. **Cost**:
   - **Definition**: The amount of money required to build, maintain, and run a data engineering system.
   - **Explanation**: When creating a data system, it's important to consider how much it will cost, including hardware, software, and maintenance expenses.
   - **Example**: If setting up a data storage system costs too much, consider a cheaper cloud solution like Amazon S3.
2. **Agility**:
   - **Definition**: The ability to quickly adapt to changes or new requirements.
   - **Explanation**: Agility means the system can be changed easily without too much effort. This helps in making quick decisions based on new data or insights.
   - **Example**: Being able to add a new feature to the data pipeline without disrupting the entire system.
3. **Scalability**:
   - **Definition**: The capacity to handle increasing amounts of data or users without performance issues.
   - **Explanation**: A scalable system can grow smoothly as data or the number of users increase, without needing a complete redesign.
   - **Example**: When your application grows from 100 to 10,000 users, the system should still run smoothly.
4. **Simplicity**:
   - **Definition**: The ease of understanding, using, and maintaining the system.
   - **Explanation**: A simple system is less likely to have errors and is easier to manage and update.
   - **Example**: Designing a database structure that is straightforward and easy for others to understand.
5. **Reuse**:
   - **Definition**: The ability to use existing components or data for new purposes.
   - **Explanation**: Reusing code or data models saves time and resources instead of building everything from scratch.
   - **Example**: Reusing an existing data processing script for a new project rather than writing a new one.
6. **Interoperability**:
   - **Definition**: The capability of different systems to work together smoothly.
   - **Explanation**: A system with good interoperability can easily connect and share data with other systems, even if they use different technologies.
   - **Example**: Ensuring a data pipeline can integrate data from both SQL and NoSQL databases.

DATA CUBE:

A **3D data cube** is like a big, three-dimensional table that helps organize and analyze data in a structured way. Think of it as a **Rubik's cube**, but instead of colored squares, each small box (or cell) inside the cube holds some kind of data or information.

How It Works:

- **Dimensions:** Just like a Rubik's cube has three dimensions (height, width, depth), a 3D data cube also has three dimensions. These dimensions can represent different things, such as **time**, **location**, and **product**.
- **Data Storage:** Each cell in this 3D cube holds a value, like sales figures or inventory levels, based on the combination of the three dimensions.

Example:

Imagine you have a data cube that shows **sales** of a product:

1. **Dimension 1:** Different months of the year (e.g., January, February, etc.)
2. **Dimension 2:** Different store locations (e.g., New York, Chicago, etc.)
3. **Dimension 3:** Different products (e.g., shoes, shirts, etc.)

With this 3D cube, you can find specific information like **"How many shoes were sold in Chicago in February?"** or compare sales across different months and locations easily.

DATA ENGINEERING LIFECYCLE:

1. **Generation**:
   - **Definition**: This is the phase where raw data is created or collected.
   - **Explanation**: Data can be generated from different sources, such as sensors, websites, or user activities. This stage is all about capturing and generating the initial data that will be processed later.
   - **Example**: Logging user activities on a website, such as page views or clicks.
2. **Ingestion**:
   - **Definition**: The process of bringing raw data into the system.
   - **Explanation**: This step involves gathering data from various sources and sending it to a storage system for further use. It includes collecting data in real-time or in batches.
   - **Example**: Collecting data from different sensors and sending it to a central data warehouse.
3. **Transformation**:
   - **Definition**: The process of changing and cleaning the data to make it usable.

- **Explanation**: Raw data often needs to be processed, cleaned, or formatted before it can be used for analysis. This step includes converting data types, removing errors, or aggregating information.
- **Example**: Converting date formats or calculating the average temperature from hourly readings.

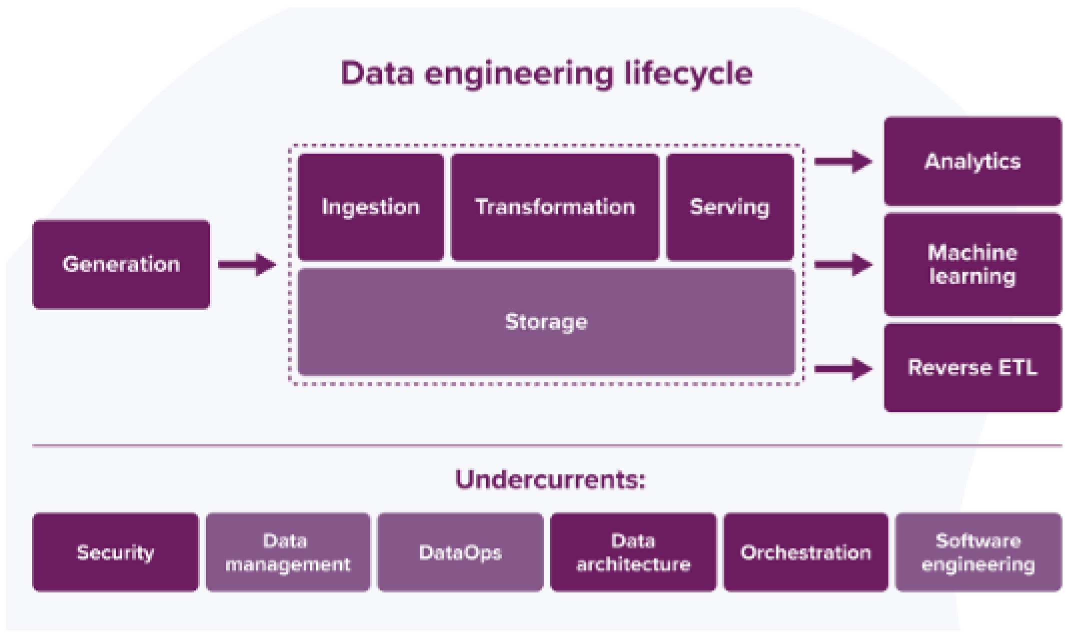
4. **Serving**:
    - **Definition**: Making the processed data available for use.
    - **Explanation**: Once the data is cleaned and transformed, it needs to be served to applications or users. This step ensures the data is accessible in the right format and structure.
    - **Example**: Providing the cleaned data to a dashboard for visualizing key metrics.
5. **Storage**:
    - **Definition**: Storing raw and processed data in databases or storage systems.
    - **Explanation**: Storage is where data is kept securely for future use. It should be designed to handle large volumes of data and allow for easy access.
    - **Example**: Storing user activity logs in a cloud storage system like Amazon S3.
6. **Analytics**:
    - **Definition**: Using data to derive insights and make decisions.
    - **Explanation**: Analytics involves examining and interpreting data to find patterns, trends, or insights. It helps organizations understand what the data means.
    - **Example**: Analyzing sales data to identify the best-selling product of the month.
7. **Machine Learning**:
    - **Definition**: Applying algorithms to data to create predictive models.
    - **Explanation**: Machine learning uses data to train models that can make predictions or categorize new data. It's a way to build smart systems.
    - **Example**: Using customer data to predict which products they might buy next.
8. **Reverse ETL (Extract, Transform, Load)**:
    - **Definition**: Sending transformed data back to operational systems.
    - **Explanation**: Reverse ETL takes the data that was analyzed and sends it back to the systems where it can be used in real-time, such as CRM tools or sales platforms.
    - **Example**: Sending customer insights from the data warehouse back to a marketing platform for targeted campaigns.

Undercurrents:

These are foundational elements that support the entire lifecycle:

1. **Security**: Ensuring data is protected and only accessible to authorized users.
2. **Data Management**: Organizing and managing data so it can be easily accessed and used.
3. **DataOps**: Practices and tools for managing data operations efficiently.
4. **Data Architecture**: Structuring how data is stored and accessed.
5. **Orchestration**: Automating workflows and processes to manage data pipelines.

6. **Software Engineering**: Developing tools and systems for efficient data processing.



Evaluating Source Systems: Key Engineering Considerations

1. **Essential Characteristics of the Data Source:**
   - What kind of source is it? (An application? A system of IoT devices?)
2. **Data Persistence in Source Systems:**
   - How is data stored? Is it stored long-term, short-term, or deleted quickly after use?
3. **Data Generation Rate:**
   - At what rate is data generated? (How many events per second? How much data in gigabytes per hour?)
4. **Error Frequency:**
   - How often do errors occur in the data?
5. **Duplicate Data:**
   - Will there be any duplicates in the data?
6. **Late Data Arrival:**
   - Are there any instances where data arrives late?
7. **Data Schema:**
   - What is the schema of the ingested data?
8. **Data Joining Across Multiple Systems:**

    - Is there a need to join data from multiple tables or systems to get a complete view?
9. **Handling Schema Changes:**
    - How are schema changes managed and communicated to downstream stakeholders?
10. **Data Extraction Frequency:**
    - How often should data be pulled from the source systems?
11. *Example: For an airplane's flight data, the state during take-off differs from the state during flight. So, schema and data in the source system may vary based on these states.*
12. **Stateful Systems:**
    - In stateful systems, is data captured through periodic snapshots or event updates? (For example, using Change Data Capture (CDC) techniques.)
    - What logic is used to track changes in the source database?

Key Engineering Considerations : Data Storage

1. **Choosing the Right Storage Solution**
    - Understand what type of storage solution is required:
        - Is it object storage like Amazon S3 (used for unstructured data like images and videos)?
        - Or is it a cloud data warehouse like Google BigQuery (used for structured data and complex queries)?
2. **Support for Complex Query Patterns**
    - Determine if the storage solution needs to support advanced querying capabilities, such as:
        - SQL-like queries.
        - Filtering data directly within storage (e.g., using Amazon S3 Select).
3. **Schema Flexibility**
    - Evaluate if the storage system is schema-agnostic (supports dynamic schema changes, like NoSQL databases) or requires a fixed schema (like traditional SQL databases).
4. **Handling Metadata**
    - Metadata is critical for understanding the stored data. It includes information about:
        - Schema evolution (how the structure of data changes over time).
        - Data flows and transformations (how data moves from one place to another).
5. **Data Governance**
    - Consider data quality, master data management, and data lineage (tracking where data comes from and how it has changed) to maintain governance and reliability.
6. **Compatibility with Data Architecture**

    - Assess whether the storage solution can integrate seamlessly with the existing data architecture:
        - Does it support the required read and write speeds?
        - Can it handle the anticipated data volume and operations?
7. **Scalability and Capacity Limits**
    - Ensure the storage system can scale to accommodate future growth:
        - Consider limitations like total storage capacity, maximum read/write operations, and throughput.
8. **Data Retrieval and Access Patterns**
    - Define how users and downstream processes will access the data:
        - Can the storage solution handle frequent reads and writes without performance degradation?
        - Does it support different retrieval patterns, such as random vs. sequential access?
9. **Service Level Agreements (SLAs)**
    - Verify if the storage system meets the required SLAs for:
        - Data availability and uptime.
        - Performance metrics, such as query response times.
10. **Ingestion and Storage Ordering**
    - The data engineering lifecycle often places ingestion before storage:
        - Understand how the storage solution handles data ingestion.
        - Ensure it supports real-time or batch ingestion as needed.
11. **Impact on Other Stages of the Data Lifecycle**
    - Storage is not an isolated stage—it impacts other stages like ingestion, transformation, and querying.
    - Choose a solution that can support these stages efficiently.
12. **Understanding Storage Technology**
    - Familiarize yourself with how the storage system works to avoid improper usage:
        - For example, using object storage (like S3) for frequent random updates can lead to poor performance.
13. **Creating a Data Lake for Downstream Processing**
    - Decide if the storage solution will serve as a data lake (a centralized repository for all types of data) to enable easier downstream processing and analytics.
14. **Capturing Metadata and Schema Changes**
    - Ensure that metadata about schema changes and data flow is captured:
    - This information helps future projects and makes it easier to adapt to changes in the data architecture.

After understanding the source of the data, its characteristics, and how it is stored, the next step in the data engineering lifecycle is *data ingestion*. Data ingestion refers to the process of bringing data from source systems into your own system for further storage and processing.

Since data ingestion involves transferring data from external sources (which are not directly under your control), several challenges can arise, such as:

- Unresponsive data sources.
- Data being delivered in poor quality or incomplete.
- Ingestion services mysteriously fail, which can halt or reduce data delivery.

Such unreliability in data ingestion can create a ripple effect throughout the data engineering lifecycle, affecting downstream processes like storage, transformation, and analysis.

Key Engineering Considerations for Data Ingestion:

1. **Understand the Use Case for the Data:**
    - What purpose will this data serve?
    - Can this data be reused for multiple purposes, or will you need to create different versions of the same data set?
2. **Reliability of the Source System:**
    - Is the system generating and delivering this data reliable?
    - Is the data available when you need it?
3. **Data Destination:**
    - Where will the ingested data be stored or processed after ingestion?
    - What format should it be in to be compatible with the destination system?
4. **Data Access Frequency:**
    - How often will you need to access this ingested data?
    - Should the ingestion process support real-time (streaming) or batch (periodic) ingestion?

Batch vs Stream Data Ingestion

Understanding the difference between **Batch** and **Stream** data ingestion is essential to selecting the right data processing strategy:

Batch Ingestion:

- **Definition:** Batch ingestion involves collecting data over a certain period and processing it in chunks or batches.
- **Example:** Imagine you have an e-commerce system that updates all its production tables with new order data every 24 hours. The data collected throughout the day is stored and then processed together at the end of the day.
- **When to Use:** Batch ingestion is ideal when real-time updates are not required. It’s suitable for scenarios where data is processed in bulk and does not need to be available immediately, such as generating daily reports.

Stream Ingestion:

- **Definition:** Stream ingestion processes data as it is generated, providing it to downstream systems in real-time or near real-time.
- **Example:** In the same e-commerce system, if you want to update the sales dashboard immediately after each sale, streaming ingestion would push new order data to the dashboard in less than a second.
- **When to Use:** Stream ingestion is useful for time-sensitive applications that require instant data availability, such as fraud detection, monitoring, or real-time analytics.

Summary:

- **Batch ingestion** is like collecting all the mail at the end of the day and delivering it together.
- **Stream ingestion** is like delivering mail piece-by-piece as it arrives, providing instant updates.

By choosing the appropriate ingestion method, you can meet your system's requirements for data freshness, processing speed, and resource optimization.

- **Push Model:**
  - **Definition:** The source system sends data to a target location such as a database, file system, or object store.
  - **Example:** Consider a real-time sensor system that automatically sends temperature data to a central database every minute.
- **Pull Model:**
  - **Definition:** The target system retrieves data from the source periodically.
  - **Example:** A database pulls customer transaction data from an e-commerce website every 30 minutes.
- **Hybrid Approach:**
  - Often, data pipelines can involve a combination of push and pull models as data moves through various stages of the pipeline, depending on the requirements and architecture of each system.

Data Transformation

Once data is ingested and stored, it typically needs to be transformed into a format that is useful for downstream applications such as reporting, analysis, or machine learning.

What is Data Transformation?

- Changing the structure, format, or schema of data.

- Removing, adding, or modifying records.
- Standardizing formats and data types.
- Cleaning or enriching data by removing errors or filling in missing values.

Purpose of Data Transformation:

Without proper transformation, data might not be in a usable form for reporting, analysis, or other applications. The aim is to refine raw data into a more structured and insightful format.

Key Considerations for Data Transformation:

1. **Cost and ROI:**
    - What are the costs associated with transforming data, and what business value does it add?
2. **Simplicity and Modularity:**
    - Keep transformations as simple and self-contained as possible. Complex or interdependent transformations can lead to increased errors and maintenance costs.
3. **Alignment with Business Rules:**
    - Ensure transformations support specific business rules or use cases.

Batch vs Streaming Transformations:

- **Batch Transformations:**
    - Involves processing large chunks of data at set intervals (e.g., hourly or daily).
    - Example: Aggregating sales data at the end of each day for reporting.
- **Streaming Transformations:**
    - Transforms data in real-time as it arrives.
    - Example: Real-time sentiment analysis of customer feedback as it is submitted.
- **Future of Streaming Transformations:**
  With the rise of real-time data processing needs and advancements in stream processing technologies, streaming transformations are becoming more popular and may replace batch processing in certain domains.

---

4 Vs of Data:

1. **Velocity**: How quickly the data is generated or changes.
2. **Veracity**: How accurate or reliable the data is.
3. **Volume**: The size of the data.
4. **Variety**: Different types of data like audio, video, images, etc.
5. **Value**: The worth of the data for the business.

Serving Data:

This is the final stage of the data engineering lifecycle. Once data is ingested, stored, and transformed into useful forms, it's time to make it available to users. Serving data involves providing this data to users or systems so it can be used for analytics, reporting, or decision-making.

- Data should have value and should not just be stored without a purpose.
- Many companies face risks with "vanity projects," where they gather huge amounts of data but don't use it in any meaningful way.

Data Transformation:

- After ingestion and storage, data might need to be transformed. Transformation means converting the raw data into a more useful format for analysis or machine learning.
- This can include changing data types, reformatting records, or removing bad data.
- Advanced transformations might include normalizing the schema, performing aggregations for reporting, or creating features for ML models.

Key Considerations:

1. What's the cost and benefit of transforming the data?
2. Is the transformation as simple and isolated as possible?
3. What business rules or logic do these transformations follow?

Machine Learning (ML):

- ML is a key part of the data lifecycle and can add significant value to data.
- Data engineers and ML engineers often work together. ML engineers may need to support specialized pipelines, large data clusters, and data orchestration.
- Data engineers need to ensure that data is well-organized, reliable, and easy to access for ML purposes.
