Types of Data

Types of Data

- Structured
- Unstructured
- Semi-Structured

1. We have
2. Structured data

Structured Data

- Definition: Data that is organized in a defined manner or schema, typically found in relational databases.
- Characteristics:
  - Easily queryable
  - Organized in rows and columns
  - Has a consistent structure
- Examples:
  - Database tables
  - CSV files with consistent columns
  - Excel spreadsheets

here we can easily write sql queries and manage the data

3. Unstructured Data

Unstructured Data

- Definition: Data that doesn't have a predefined structure or schema.
- Characteristics:
  - Not easily queryable without preprocessing
  - May come in various formats
- Examples:
  - Text files without a fixed format
  - Videos and audio files
  - Images
  - Emails and word processing documents

to extract the data…we need to process the files first..for example for video and audio files…we can get metadata like data created, duration, summary etc…by processing them

4. SemiStructured Data

Semi-Structured Data

- Definition: Data that is not as organized as structured data but has some level of structure in the form of tags, hierarchies, or other patterns.
- Characteristics:
  - Elements might be tagged or categorized in some way
  - More flexible than structured data but not as chaotic as unstructured data
- Examples:
  - XML and JSON files
  - Email headers (which have a mix of structured fields like date, subject, etc., and unstructured data in the body)
  - Log files with varied formats

Here are some examples of semi-structured data:

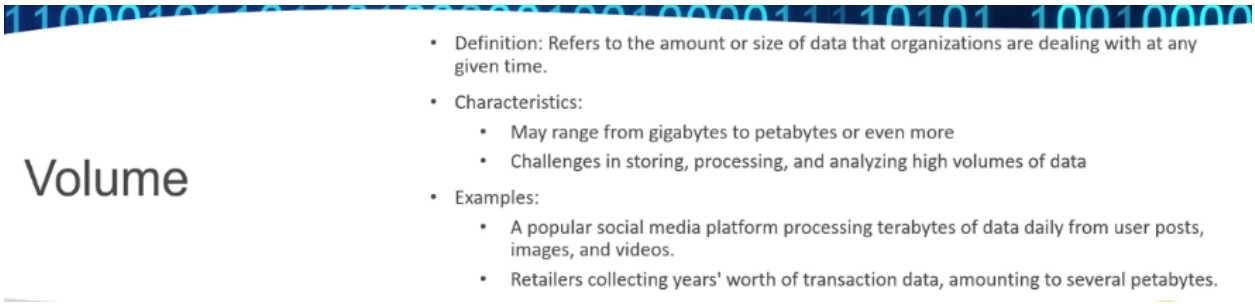
- **Email:** An email contains structured elements like sender name, recipient address, subject line, date, and timestamps. It also includes unstructured text content within the body of the email.
- **Log files:** Log files often contain a mix of structured data like timestamps, IP addresses, and error codes, along with unstructured messages describing events.
- **Social media posts:** A social media post typically includes a structured element like the author and timestamp, and unstructured content like the text of the post, images, and embedded links.
- **JSON (JavaScript Object Notation) data:** JSON is a common format for exchanging data between applications. It uses key-value pairs to store data, providing a hierarchical structure, but the data within each key can be flexible.
- **XML (Extensible Markup Language) data:** XML uses tags to define the structure of data. XML documents can include both structured elements with defined tags and unstructured content.

Properties of data

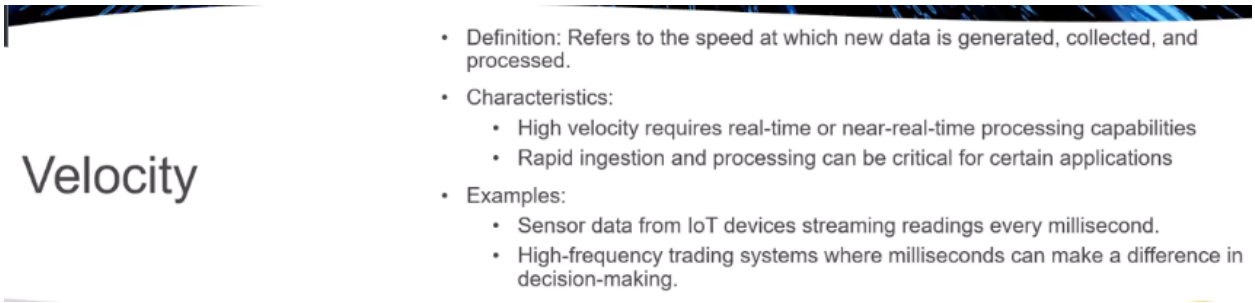
Properties of Data

- Volume
- Velocity
- Variety

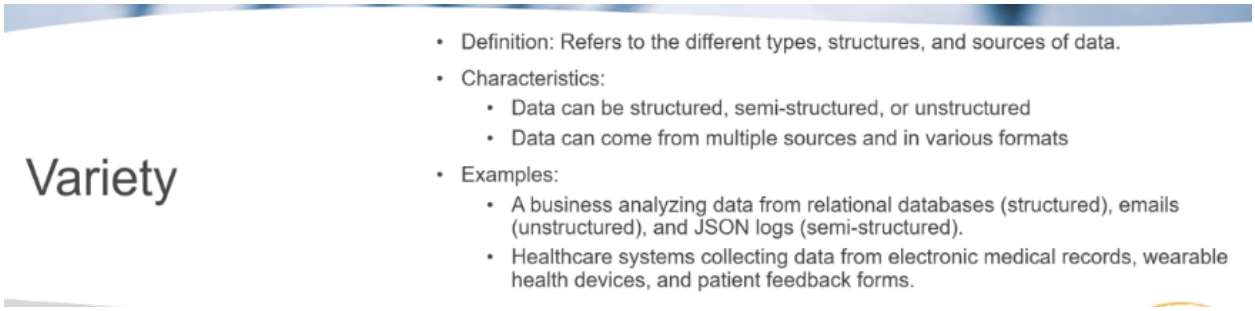
1.
2. Volume



3. Velocity



4. Variety



7. Data Warehouses vs Data Lakes

1. Data Warehouses

Data Warehouse

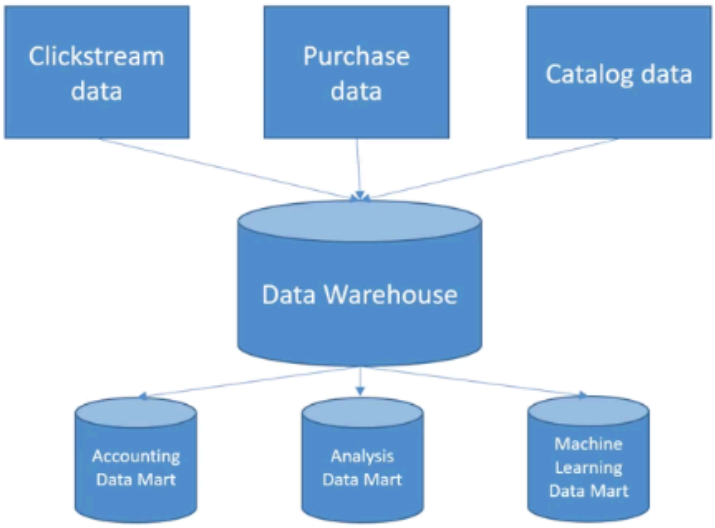
- Definition: A centralized repository optimized for analysis where data from different sources is stored in a structured format.
- Characteristics:
  - Designed for complex queries and analysis
  - Data is cleaned, transformed, and loaded (ETL process)
  - Typically uses a star or snowflake schema
  - Optimized for read-heavy operations
- Examples:
  - Amazon Redshift
  - Google BigQuery
  - Microsoft Azure SQL Data Warehouse

Amazon Redshift

Here the data will be coming from different sources and the data warehouse organizes the data to be queries efficiently.

2. Example of Datawarehouse(Amazon)

Data Warehouse example

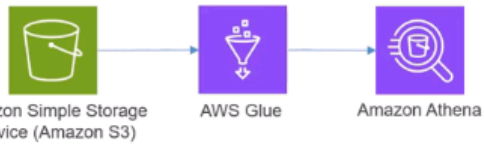


here data is getting from different sources ….like clicks,purchases etc…and after the we have created views(DataMart) from this data…as per the teams need

3. DataLake

Data Lake

- Definition: A storage repository that holds vast amounts of raw data in its native format, including structured, semi-structured, and unstructured data.
- Characteristics:
  - Can store large volumes of raw data without predefined schema
  - Data is loaded as-is, no need for preprocessing
  - Supports batch, real-time, and stream processing
  - Can be queried for data transformation or exploration purposes
- Examples:
  - Amazon Simple Storage Service (S3) when used as a data lake
  - Azure Data Lake Storage
  - Hadoop Distributed File System (HDFS)



In datalake we load the data as it is…and then extract the structure using any tools..so here we are just storing the raw data

4. In AWS..we use glue to extract the structure and schema from that unstructured data that's sitting in S3.
5. Comparing the two

Comparing the two

- **Schema:**
  - Data Warehouse: Schema-on-write (predefined schema before writing data)
    - Extract – Transform – Load (**ETL**)
  - Data Lake: Schema-on-read (schema is defined at the time of reading data)
    - Extract – Load – Transform (**ELT**)
- **Data Types:**
  - Data Warehouse: Primarily structured data
  - Data Lake: Both structured and unstructured data
- **Agility:**
  - Data Warehouse: Less agile due to predefined schema
  - Data Lake: More agile as it accepts raw data without a predefined structure
- **Processing:**
  - Data Warehouse: ETL (Extract, Transform, Load)
  - Data Lake: ELT (Extract, Load, Transform) or just Load for storage purposes
- **Cost:**
  - Data Warehouse: Typically more expensive because of optimizations for complex queries
  - Data Lake: Cost-effective storage solutions, but costs can rise when processing large amounts of data

6. More on agility

- **Data Warehouse:** A data warehouse is a repository that stores large amounts of structured data, typically from multiple sources. The data is transformed and organized according to a predefined schema, which is essentially a blueprint that defines the data's structure. This schema design allows for efficient querying and analysis of the data. However, it can also make data warehouses less agile, or adaptable, to new data sources or business needs. If the schema doesn't anticipate a new data point, it can be difficult to incorporate it into the data warehouse.

- **Data Lake:** In contrast, a data lake is a repository that stores large amounts of raw data in its original format. There is no predefined schema, so the data can come from any source and doesn't need to be structured in a specific way. This makes data lakes more agile and flexible, but it can also make it more challenging to query and analyze the data.

Here's an analogy to help understand the difference:

- Think of a data warehouse like a traditional library. The books are neatly categorized and shelved according to a specific system (the schema). This makes it easy to find exactly the book you're looking for, but it can be time-consuming to add new books if they don't fit neatly into an existing category.

- A data lake, on the other hand, would be more like a big pile of books. You can toss any book in there, regardless of genre or publication date. This makes it easy to add new information, but it can be a challenge to find exactly what you're looking for without some digging.

7.

Choosing a Warehouse vs. a Lake

- **Use a Data Warehouse when:**
  - You have structured data sources and require fast and complex queries.
  - Data integration from different sources is essential.
  - Business intelligence and analytics are the primary use cases.
- **Use a Data Lake when:**
  - You have a mix of structured, semi-structured, or unstructured data.
  - You need a scalable and cost-effective solution to store massive amounts of data.
  - Future needs for data are uncertain, and you want flexibility in storage and processing.
  - Advanced analytics, machine learning, or data discovery are key goals.
- Often, organizations use a combination of both, ingesting raw data into a data lake and then processing and moving refined data into a data warehouse for analysis.

8. Data Lakehouse

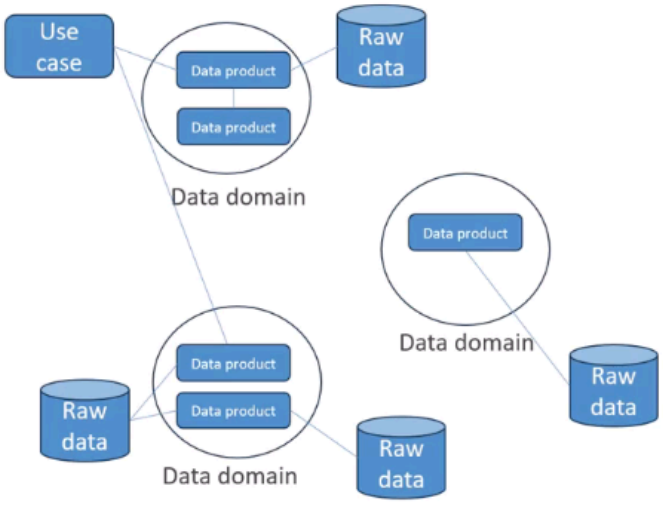
Data Lakehouse

- Definition: A hybrid data architecture that combines the best features of data lakes and data warehouses, aiming to provide the performance, reliability, and capabilities of a data warehouse while maintaining the flexibility, scale, and low-cost storage of data lakes.
- Characteristics:
  - Supports both structured and unstructured data.
  - Allows for schema-on-write and schema-on-read.
  - Provides capabilities for both detailed analytics and machine learning tasks.
  - Typically built on top of cloud or distributed architectures.
  - Benefits from technologies like Delta Lake, which bring ACID transactions to big data.
- Examples:
  - **AWS Lake Formation (with S3 and Redshift Spectrum)**
  - **Delta Lake:** An open-source storage layer that brings ACID transactions to Apache Spark and big data workloads.
  - **Databricks Lakehouse Platform:** A unified platform that combines the capabilities of data lakes and data warehouses.
  - **Azure Synapse Analytics:** Microsoft's analytics service that brings together big data and data warehousing.

What is Data Mesh?

Data Mesh

- Coined in 2019; it's more about governance and organization
- Individual teams own "data products" within a given domain
- These data products serve various "use cases" around the organization
- "Domain-based data management"
- Federated governance with central standards
- Self-service tooling & infrastructure
- Data lakes, warehouses, etc. may be part of it
  - But a "data mesh" is more about the "data management paradigm" and not the specific technologies or architectures.



1.

Example: E-commerce Data Mesh

Let's consider an e-commerce company with Data Mesh:

- **Domains:** Marketing, Sales, Product, and Order Fulfillment could be individual domains.
- **Marketing Data:** The marketing domain might own data about customer behavior on the website, email campaigns, and social media engagement. They clean and transform this data into usable metrics and customer segments. This data product could be a "Customer Engagement Dashboard" accessible to other domains through a well-defined API.
- **Sales Data:** The sales domain might manage customer order history, lead scoring data, and contact information. They could create a "Sales Opportunity Report" data product for marketing to identify high-potential customers.
- **Benefits:** This approach allows each domain to be agile and adapt their data products quickly to changing business needs. It also fosters data ownership and accountability within each domain, leading to higher quality data.

Data Mesh vs. Data Lake

While data lakes store all data in one place, a Data Mesh focuses on ownership and organized access by domain. Think of a data lake as a giant warehouse with everything mixed together, while a Data Mesh is like a well-organized store with dedicated departments (domains) managing their products (data).

Managing and Extracting DataPipelines

1. Lets look at extract

   - **Extract:**
     - Retrieve raw data from source systems, which can be databases, CRMs, flat files, APIs, or other data repositories.
     - Ensure data integrity during the extraction phase.
     - Can be done in real-time or in batches, depending on requirements.

   data can from API's, ..and we need to make sure the data is not corrupted while extracting
2. Transform

   ### ETL Pipelines: Transform

   - Convert the extracted data into a format suitable for the target data warehouse.
   - Can involve various operations such as:
     - Data cleansing (e.g., removing duplicates, fixing errors)
     - Data enrichment (e.g., adding additional data from other sources)
     - Format changes (e.g., date formatting, string manipulation)
     - Aggregations or computations (e.g., calculating totals or averages)
     - Encoding or decoding data
     - Handling missing values

3. Loading

ETL Pipelines: Load

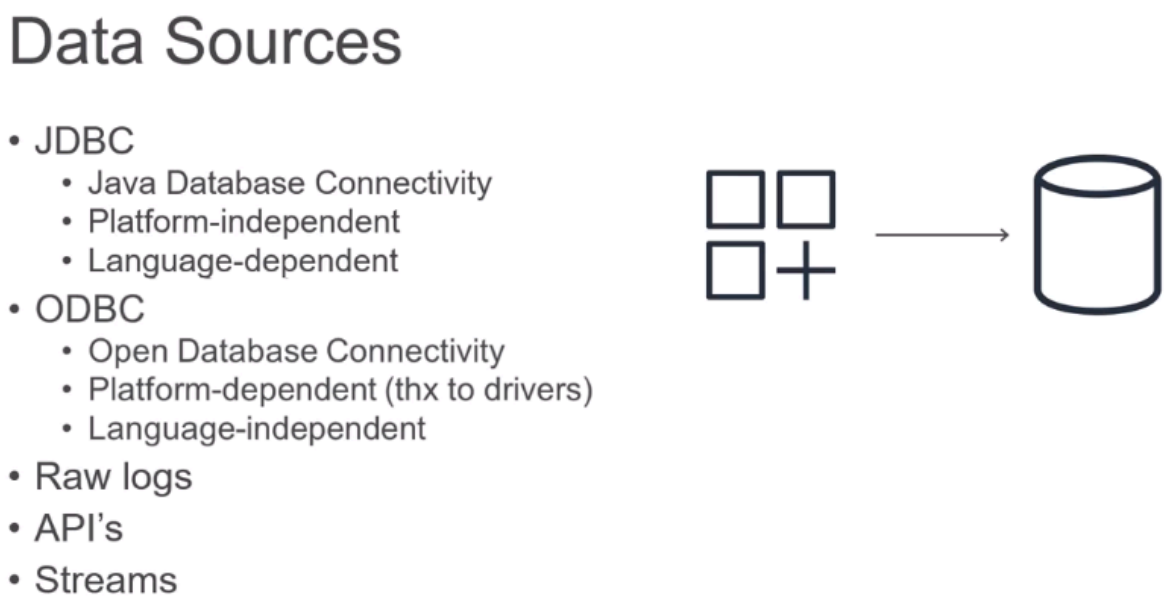
- Move the transformed data into the target data warehouse or another data repository.
- Can be done in batches (all at once) or in a streaming manner (as data becomes available).
- Ensure that data maintains its integrity during the loading phase.

4. Managing ETL pipelines

Managing ETL Pipelines

- This process must be automated in some reliable way
- AWS Glue
- Orchestration services
  - EventBridge
  - Amazon Managed Workflows for Apache Airflow [Amazon MWAA]
  - AWS Step Functions
  - Lambda
  - Glue Workflows
- We'll get into specific architectures later.

Common DataSources and Data Formats

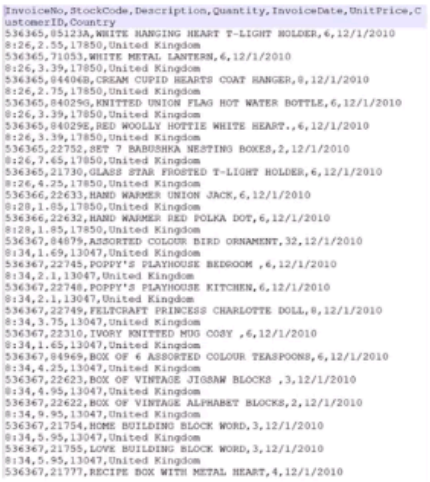


1.
2. Data Sources Explained : https://g.co/gemini/share/dc6d200d32d4
3. Data Formats

4. csv

CSV (Comma-Separated Values)

- **Description:** Text-based format that represents data in a tabular form where each line corresponds to a row and values within a row are separated by commas.
- **When to Use:**
  - For small to medium datasets.
  - For data interchange between systems with different technologies.
  - For human-readable and editable data storage.
  - Importing/Exporting data from databases or spreadsheets.
- **Systems:** Databases (SQL-based), Excel, Pandas in Python, R, many ETL tools.



```
InvoiceNo,StockCode,Description,Quantity,InvoiceDate,UnitPrice,C
ustomerID,Country
536365,85123A,WHITE HANGING HEART T-LIGHT HOLDER,6,12/1/2010
8:26,2.55,17850,United Kingdom
536365,71053,WHITE METAL LANTERN,6,12/1/2010
8:26,3.39,17850,United Kingdom
536365,84406B,CREAM CUPID HEARTS COAT HANGER,8,12/1/2010
8:26,2.75,17850,United Kingdom
536365,84029G,KNITTED UNION FLAG HOT WATER BOTTLE,6,12/1/2010
8:26,3.39,17850,United Kingdom
536365,84029E,RED WOOLLY HOTTIE WHITE HEART.,6,12/1/2010
8:26,3.39,17850,United Kingdom
536365,22752,SET 7 BABUSHKA NESTING BOXES,2,12/1/2010
8:26,7.65,17850,United Kingdom
536365,21730,GLASS STAR FROSTED T-LIGHT HOLDER,6,12/1/2010
8:26,4.25,17850,United Kingdom
536366,22633,HAND WARMER UNION JACK,6,12/1/2010
8:28,1.85,17850,United Kingdom
536366,22632,HAND WARMER RED POLKA DOT,6,12/1/2010
8:28,1.85,17850,United Kingdom
536367,84879,ASSORTED COLOUR BIRD ORNAMENT,32,12/1/2010
8:34,1.69,13047,United Kingdom
536367,22745,POPPY'S PLAYHOUSE BEDROOM ,6,12/1/2010
8:34,2.1,13047,United Kingdom
536367,22748,POPPY'S PLAYHOUSE KITCHEN,6,12/1/2010
8:34,2.1,13047,United Kingdom
536367,22749,FELTCRAFT PRINCESS CHARLOTTE DOLL,8,12/1/2010
8:34,3.75,13047,United Kingdom
536367,22310,IVORY KNITTED MUG COSY ,6,12/1/2010
8:34,1.65,13047,United Kingdom
536367,84969,BOX OF 6 ASSORTED COLOUR TEASPOONS,6,12/1/2010
8:34,4.25,13047,United Kingdom
536367,22623,BOX OF VINTAGE JIGSAW BLOCKS ,3,12/1/2010
8:34,4.95,13047,United Kingdom
536367,22622,BOX OF VINTAGE ALPHABET BLOCKS,2,12/1/2010
8:34,9.95,13047,United Kingdom
536367,21754,HOME BUILDING BLOCK WORD,3,12/1/2010
8:34,5.95,13047,United Kingdom
536367,21755,LOVE BUILDING BLOCK WORD,3,12/1/2010
8:34,5.95,13047,United Kingdom
536367,21777,RECIPE BOX WITH METAL HEART,4,12/1/2010
```

not only comma..we can use pipe too for separating the values

5. JSON

JSON (JavaScript Object Notation)

- **Description:** Lightweight, text-based, and human-readable data interchange format that represents structured or semi-structured data based on key-value pairs.
- **When to Use:**
  - Data interchange between a web server and a web client.
  - Configurations and settings for software applications.
  - Use cases that need a flexible schema or nested data structures.
- **Systems:** Web browsers, many programming languages (like JavaScript, Python, Java, etc.), RESTful APIs, NoSQL databases (like MongoDB).

```
{"messages": [{"role": "system", "content": "Data is an android
in the TV series Star Trek: The Next Generation."}, {"role":
"user", "content": "PICARD: You will agree, Data, that
Starfleet's instructions are difficult?"}, {"role":
"assistant", "content": "DATA: Difficult ... how so? Simply
solve the mystery of Farpoint Station."}]}
{"messages": [{"role": "system", "content": "Data is an android
in the TV series Star Trek: The Next Generation."}, {"role":
"user", "content": "PICARD: The problem, Data, is that another
life form built that base. How do I negotiate a friendly
agreement for Starfleet to use it while at the same time snoop
around finding how and why they built it?"}, {"role":
"assistant", "content": "DATA: Inquiry ... the word snoop ...
?"}]}
{"messages": [{"role": "system", "content": "Data is an android
in the TV series Star Trek: The Next Generation."}, {"role":
"user", "content": "PICARD: Data, how can you be programmed as
a virtual encyclopedia of human information without knowing a
simple word like snoop?"}, {"role": "assistant", "content":
"DATA: Possibility ... a kind of human behavior I was not
designed to emulate?"}]}
{"messages": [{"role": "system", "content": "Data is an android
in the TV series Star Trek: The Next Generation."}, {"role":
"user", "content": "PICARD: It means 'to spy, to sneak' ..."},
{"role": "assistant", "content": "DATA: Ah! To seek covertly,
to go stealthily, to slink, slither ..."}]}
{"messages": [{"role": "system", "content": "Data is an android
in the TV series Star Trek: The Next Generation."}, {"role":
"user", "content": "PICARD: Exactly, yes ..."}, {"role":
"assistant", "content": "DATA: ... to glide, creep, skulk,
pussyfoot, gumshoe ..."}]}
{"messages": [{"role": "system", "content": "Data is an android
in the TV series Star Trek: The Next Generation."}, {"role":
"user", "content": "WORF: Something strange on the detector
circuits ..."}, {"role": "assistant", "content": "DATA: It
registers as solid, Captain..."}]}
{"messages": [{"role": "system", "content": "Data is an android
in the TV series Star Trek: The Next Generation."}, {"role":
"user", "content": "PICARD: Records search, Data. Results of
detaching the Saucer Section at high warp speeds."}, {"role":
"assistant", "content": "DATA: Inadvisable at any warp speed,
```

6.

7. AVRO

Avro

- **Description:** Binary format that stores both the data and its schema, allowing it to be processed later with different systems without needing the original system's context.
- **When to Use:**
  - With big data and real-time processing systems.
  - When schema evolution (changes in data structure) is needed.
  - Efficient serialization for data transport between systems.
- **Systems:** Apache Kafka, Apache Spark, Apache Flink, Hadoop ecosystem.

Use Cases for Avro:

- **Data Warehousing:** Avro is a popular choice for storing data in data warehouses due to its efficiency and flexibility. It allows data to be loaded and queried quickly, making it suitable for large-scale data analysis.
- **Data Lake Storage:** Avro's schema evolution capabilities make it well-suited for data lakes, where data may not have a predefined structure initially. The schema can be added or updated later as the data matures.
- **Data Exchange:** Avro is a good option for exchanging data between different systems because it's self-describing and can be easily processed by programs written in various languages.

Here's an analogy to understand Avro:

Imagine storing your clothes. Traditional formats like JSON or XML would be like folding your clothes and storing them in large boxes, taking up a lot of space. Avro would be like vacuum-packing your clothes along with a label describing each item (schema), making them more compact and easier to identify later, even if you decide to add new clothes to the bag (schema evolution).

8. Parquet

So because it is storing data by columns instead of rows, it's optimized for analytics where your queries might only be looking at specific columns of

information and not everything that's in a row, right?

Parquet

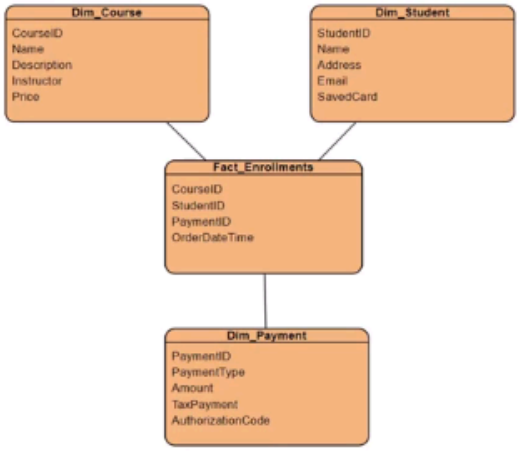
- **Description:** Columnar storage format optimized for analytics. Allows for efficient compression and encoding schemes.
- **When to Use:**
  - Analyzing large datasets with analytics engines.
  - Use cases where reading specific columns instead of entire records is beneficial.
  - Storing data on distributed systems where I/O operations and storage need optimization.
- **Systems:** Hadoop ecosystem, Apache Spark, Apache Hive, Apache Impala, Amazon Redshift Spectrum.

Basic Data Modeling, Data Lineage, Schema Evolution

1. Star Schema data model

A Very (intentionally) Incomplete Overview of Data Modeling

- The exam guide doesn't really talk about specific data models.
- But here's a star schema.
  - Fact tables
  - Dimensions
  - Primary / foreign keys
- This sort of diagram is an Entity Relationship Diagram (ERD)



2. Data Lineage

Data Lineage

- **Description:** A visual representation that traces the flow and transformation of data through its lifecycle, from its source to its final destination.
- **Importance:**
  - Helps in tracking errors back to their source.
  - Ensures compliance with regulations.
  - Provides a clear understanding of how data is moved, transformed, and consumed within systems.

3. Schema Evolution

Schema Evolution

- **Description:** The ability to adapt and change the schema of a dataset over time without disrupting existing processes or systems.
- **Importance:**
  - Ensures data systems can adapt to changing business requirements.
  - Allows for the addition, removal, or modification of columns/fields in a dataset.
  - Maintains backward compatibility with older data records.
- **Glue Schema Registry**
  - Schema discovery, compatibility, validation, registration...

```
{
 "namespace": "Customer.avro",
 "type": "record",
 "name": "Customer",
 "fields": [
 {"name": "first_name", "type": "string"},
 {"name": "last_name", "type": "string"},
 {"name": "full_name", "type": ["string", "null"], "default": null}
]
}
```

Practical Use Case: Streaming Data Pipeline

Imagine you're building a real-time analytics pipeline that processes data streams from various sources like clickstream data, social media feeds, and sensor readings. This data might arrive in different formats (JSON, Avro, etc.). Here's how Glue Schema Registry can help:

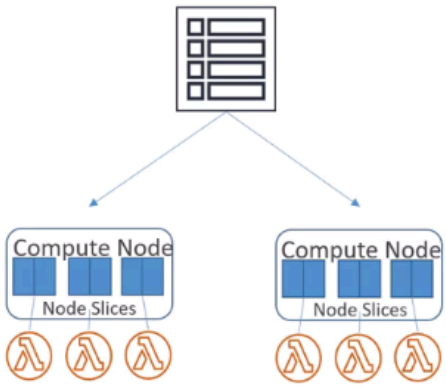
1. **Register Schemas:** You would register the schemas for each data format used in your pipeline in the Glue Schema Registry. This ensures a central definition for all data sources.
2. **Data Processing:** As data streams arrive, your processing application can retrieve the relevant schema from the registry. This schema information allows the application to correctly interpret the data structure and extract meaningful insights.
3. **Schema Evolution:** Over time, the data format coming from a particular source might change slightly. You can update the schema definition in the registry to reflect these changes. Glue Schema Registry can handle schema evolution and ensure your pipeline continues to process the data accurately.

By using Glue Schema Registry, you can build a more robust and reliable data pipeline that can handle various data formats and schema changes effectively.

Database Performance Optimization

Database Performance Optimization

- Indexing
  - Avoid full table scans!
  - Enforce data uniqueness and integrity
- Partitioning
  - Reduce amount of data scanned
  - Helps with data lifecycle management
  - Enables parallel processing
- Compression
  - Speed up data transfer, reduce storage & disk reads
  - GZIP, LZOP, BZIP2, ZSTD (Redshift examples)
    - Various tradeoffs between compression & speed
  - Columnar compression

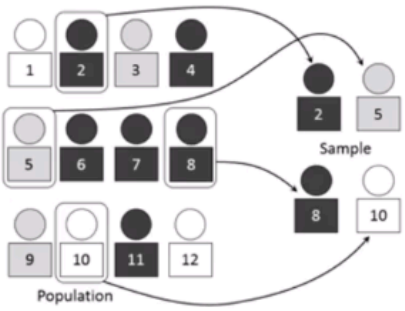


1. parquet is the example of columnar compression

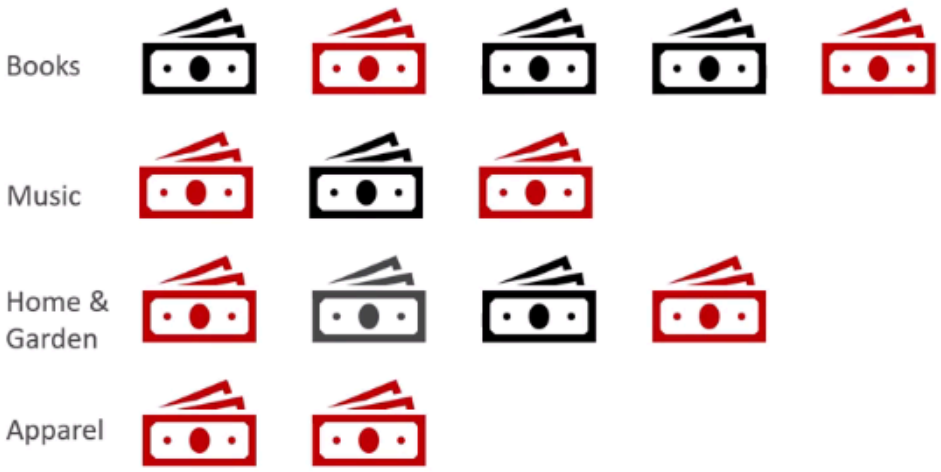
2. Data Sampling Techniques

Data Sampling Techniques

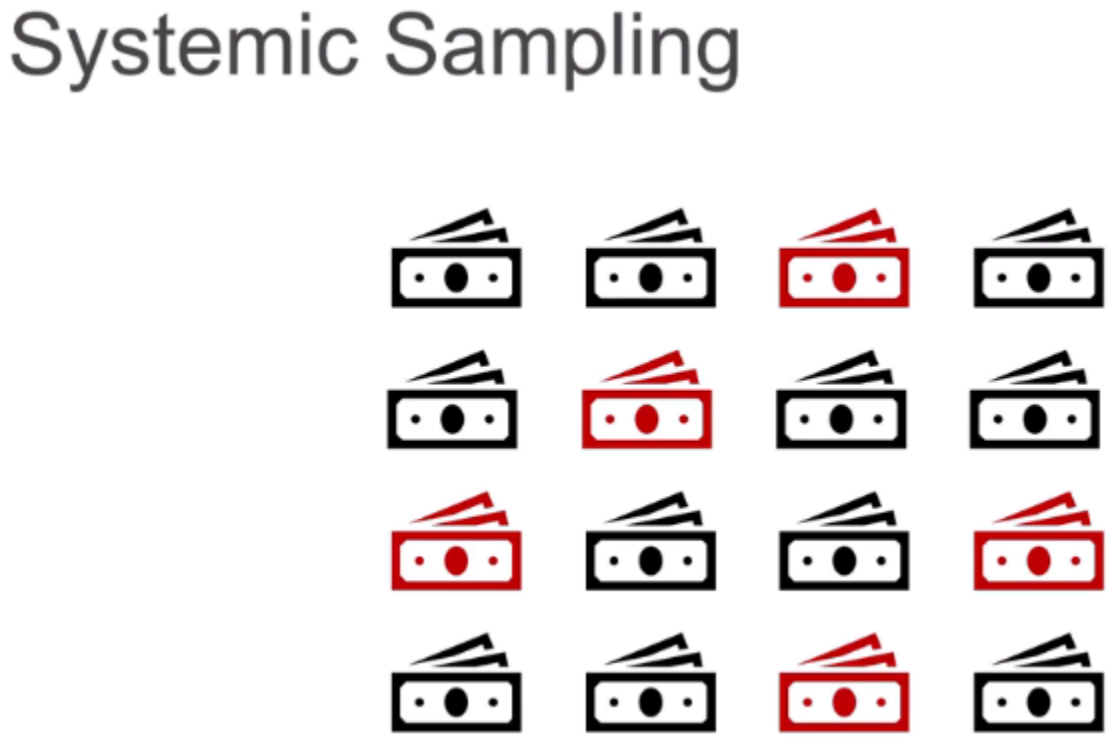
- Random Sampling
  - Everything has an equal chance
- Stratified Sampling
  - Divide population into homogenous subgroups (strata)
  - Random sample within each stratum
  - Ensures representation of each subgroup
- Others
  - Systemic, Cluster, Convenience, Judgmental



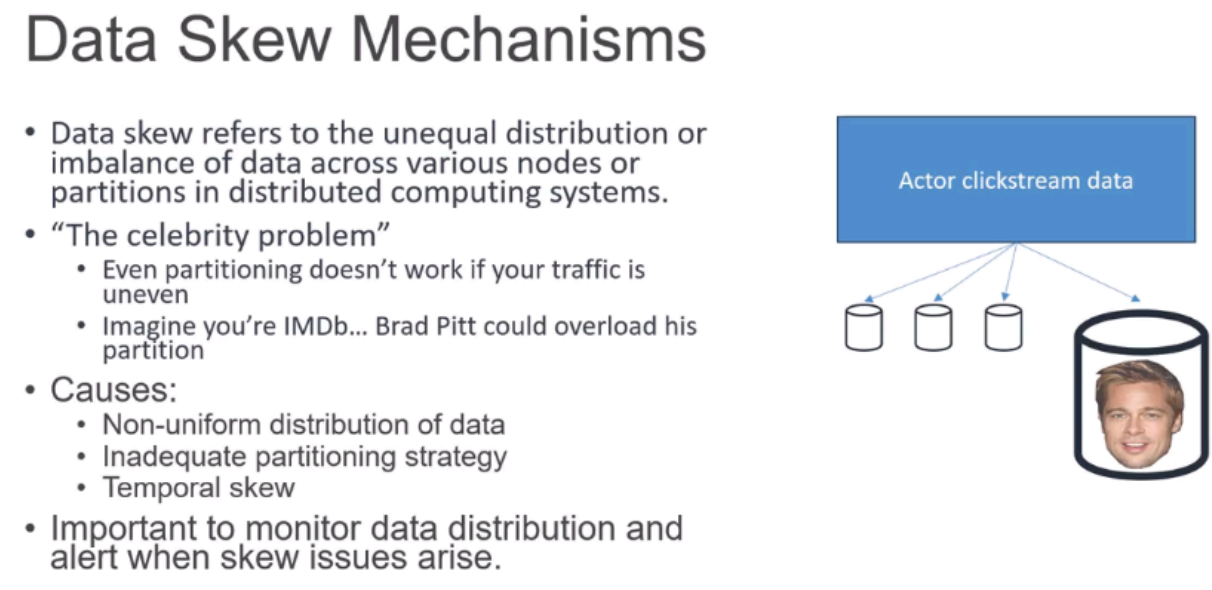
Stratified Sampling



3. In systematic Sampling…we pick every 3rd one



Data Skew Mechanisms



1.

2. Temporal skew

   Data analysis: Temporal skew can also refer to a situation where data points are not evenly distributed across time. For instance, social media activity might be much higher during evenings and weekends than during work hours. This skews the data towards certain time periods and needs to be considered when analyzing trends or drawing conclusions. Think of a graph tracking website traffic throughout the day. If most users visit at night, the graph would be skewed towards those hours, misrepresenting overall traffic distribution.

3. Addressing Data Skew

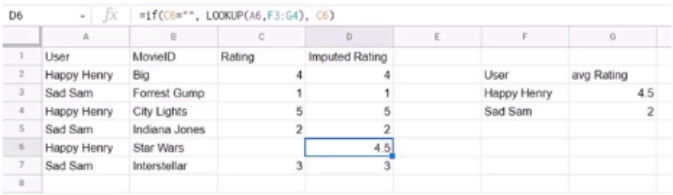
Addressing Data Skew

1. **Adaptive Partitioning:** Dynamically adjust partitioning based on data characteristics to ensure a more balanced distribution.
2. **Salting:** Introduce a random factor or "salt" to the data to distribute it more uniformly.
3. **Repartitioning:** Regularly redistribute the data based on its current distribution characteristics.
4. **Sampling:** Use a sample of the data to determine the distribution and adjust the processing strategy accordingly.
5. **Custom Partitioning:** Define custom rules or functions for partitioning data based on domain knowledge.

Data Validation and Profiling

Data Validation and Profiling

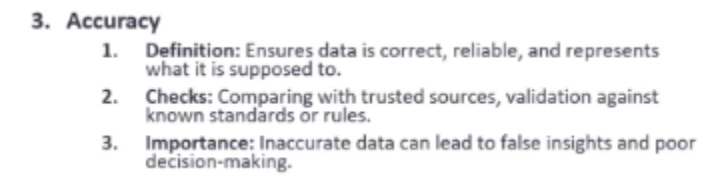
1. **Completeness**
   1. **Definition:** Ensures all required data is present and no essential parts are missing.
   2. **Checks:** Missing values, null counts, percentage of populated fields.
   3. **Importance:** Missing data can lead to inaccurate analyses and insights.
2. **Consistency**
   1. **Definition:** Ensures data values are consistent across datasets and do not contradict each other.
   2. **Checks:** Cross-field validation, comparing data from different sources or periods.
   3. **Importance:** Inconsistent data can cause confusion and result in incorrect conclusions.



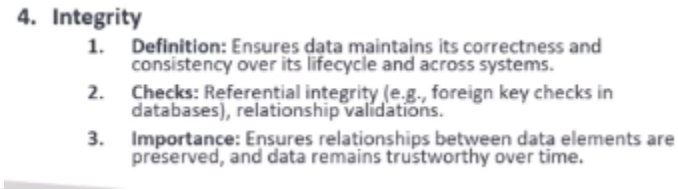
	A	B	C	D	E	F	G
1	User	MovieID	Rating	Imputed Rating			
2	Happy Henry	Big	4	4		User	avg Rating
3	Sad Sam	Forrest Gump	1	1		Happy Henry	4.5
4	Happy Henry	City Lights	5	5		Sad Sam	2
5	Sad Sam	Indiana Jones	2	2			
6	Happy Henry	Star Wars		4.5			
7	Sad Sam	Interstellar	3	3			
8							

1.

2. In consistency What if we have one table where that rating was 1 to 5 stars and we have another table where it's 1 to 10 stars? Those are not consistent rating value ranges, right? So we have to make sure they r consistent
3. Accuracy

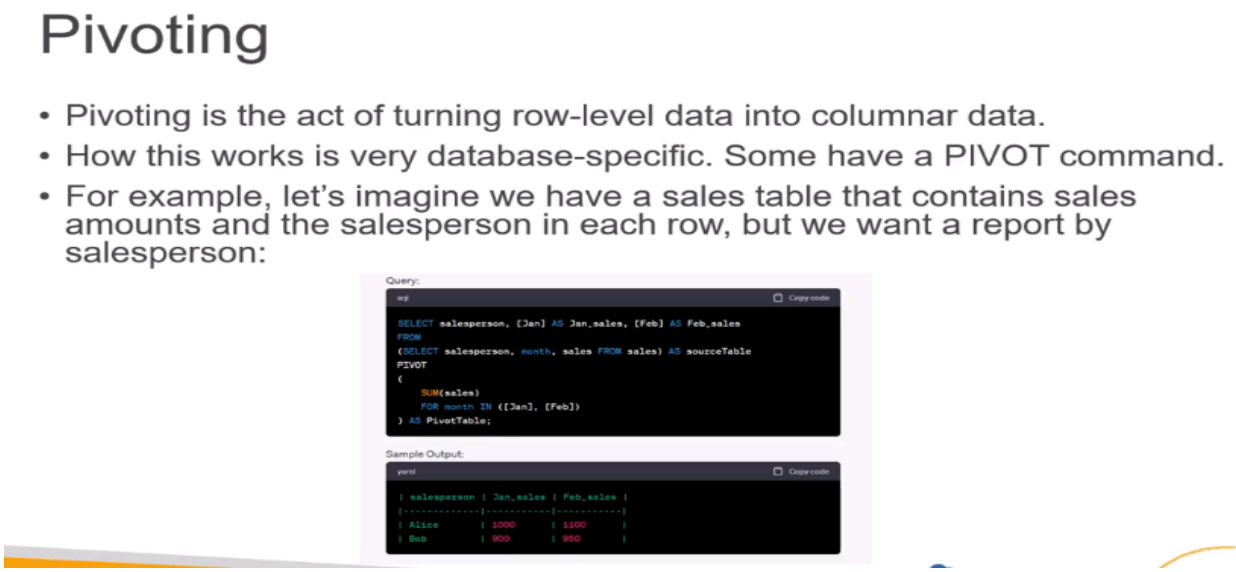


4. Integrity



5. So data validation and profiling, we need to think about data completeness.Do I have missing data? Data consistency. Is my data being represented in a consistent format across different tables?

SQL Review(Just refer online)



1.

Pivoting

- The same thing could be achieved with conditional aggregation, without requiring a specific PIVOT operation:

Query:

```sql
SELECT
    salesperson,
    SUM(CASE WHEN month = 'Jan' THEN sales ELSE 0 END) AS Jan_sales,
    SUM(CASE WHEN month = 'Feb' THEN sales ELSE 0 END) AS Feb_sales
FROM sales
GROUP BY salesperson;
```

Sample Output:

```yaml
| salesperson | Jan_sales | Feb_sales |
|-------------|-----------|-----------|
| Alice       | 1000      | 1100      |
| Bob         | 900       | 950       |
```

2.

SQL Regular Expressions

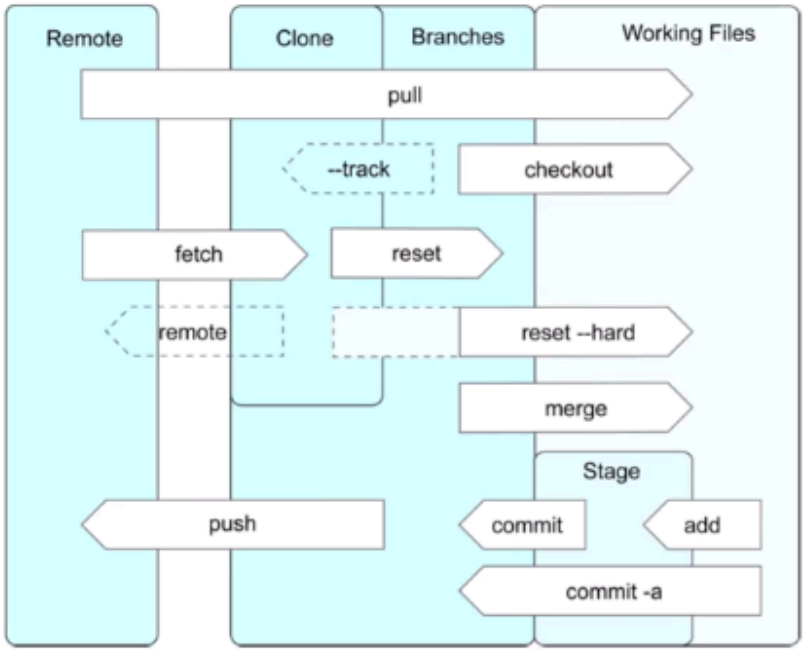
SQL Regular Expressions

- Pattern matching
  - Think a much more powerful “LIKE”
- ~ is the regular expression operator
- ~* is case-insensitive
- !~* would mean “not match expression, case insensitive”
- Regular expression 101
  - ^ - match a pattern at the start of a string
  - $ - match a pattern at the end of a string (boo$ would match boo but not book)
  - | - alternate characters (sit|sat matches both sit and sat)
  - Ranges ([a-z] matches any lower case letter)
  - Repeats ([a-z]{4} matches any four-letter lowercase word)
  - Special metacharacters
    - \d – any digit; \w – any letter, digit, or underscore, \s – whitespace, \t – tab
- Example:
  - SELECT * FROM name WHERE name ~ * ‘^(fire|ice)’;
  - Selects any rows where the name starts with “fire” or “ice” (case insensitive)

1.

Git Review

Git review



1.
2. Common git commands

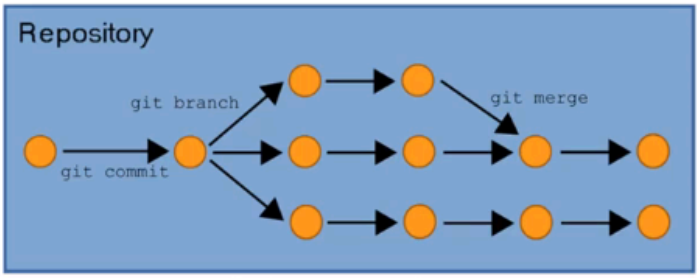
Common git commands

- Setting Up and Configuration:
  - **git init**: Initialize a new Git repository.
  - **git config**: Set configuration values for user info, aliases, and more.
    - git config --global user.name "Your Name": Set your name.
    - git config --global user.email "your@email.com": Set your email.
- Basic Commands:
  - **git clone** <repository>: Clone (or download) a repository from an existing URL.
  - **git status**: Check the status of your changes in the working directory.
  - **git add** <filename>: Add changes in the file to the staging area.
    - git add .: Add all new and changed files to the staging area.
  - **git commit** -m "Commit message here": Commit the staged changes with a message.
  - **git log**: View commit logs.

3. Git branch

Branching with git

- **git branch**: List all local branches.
  - git branch <branchname>: Create a new branch.
- **git checkout** <branchname>: Switch to a specific branch.
  - git checkout -b <branchname>: Create a new branch and switch to it.
- **git merge** <branchname>: Merge the specified branch into the current branch.
- **git branch -d** <branchname>: Delete a branch.



Felix Dreissig, noris network AG

Remote repositories

- **git remote add** <name> <url>: Add a remote repository.
- **git remote**: List all remote repositories.
- **git push** <remote> <branch>: Push a branch to a remote repository.
- **git pull** <remote> <branch>: Pull changes from a remote repository branch into the current local branch.

4.

Undoing changes

- **git reset**: Reset your staging area to match the most recent commit, without affecting the working directory.
- **git reset --hard**: Reset the staging area and the working directory to match the most recent commit.
- **git revert <commit>:** Create a new commit that undoes all of the changes from a previous commit.

5.