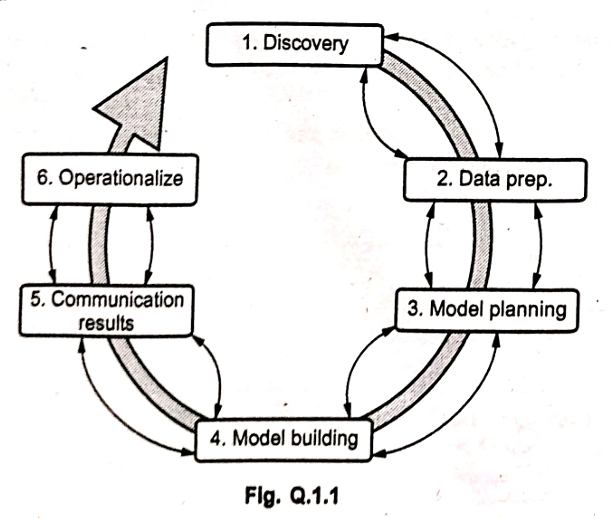
UNIT 4

Q1) Explain different steps in data analytics project life cycle .  
   
The data analytic lifecycle is a structured model designed for tackling Big Data problems and managing data science projects effectively. It consists of six phases, which can sometimes occur simultaneously, forming a cyclical process as shown in "Fig. Q.1.1 data analytic life cycle model".

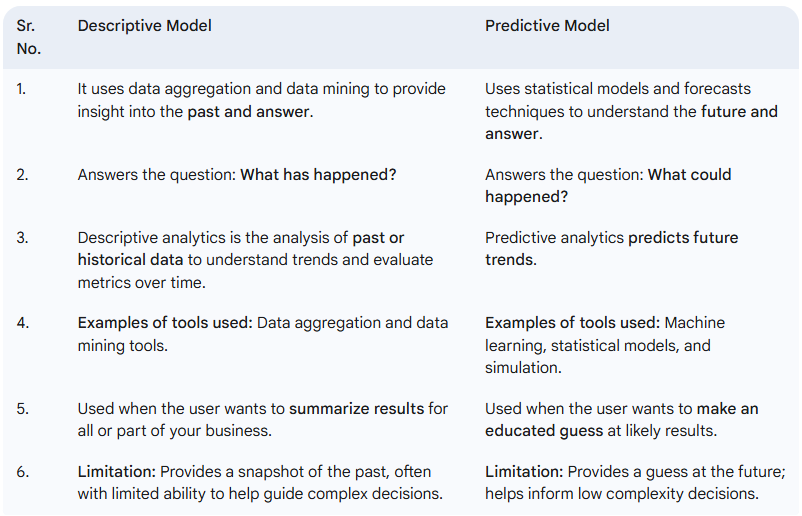
Here are the different steps (phases) in the data analytics project life cycle:  
1. **Discovery (Phase 1):**  
In this initial phase, the team focuses on understanding the business domain, including relevant history and whether the organization has attempted similar projects before.  
The goal is to learn from past experiences and leverage existing knowledge.  
The team assesses the resources available to support the project, considering people, technology, time constraints, and available data.  
2. **Data Preparation (Phase 2):**  
Phase 2 necessitates the presence of an analytic sandbox – a dedicated environment where the team can work with data and perform analytics throughout the project's duration.  
The core task in this phase is to get the data into this sandbox, which involves executing Extract, Load, and Transform (ELT) or Extract, Transform, and Load (ETL) processes.  
This phase includes data cleaning, transformation, integration, and structuring to prepare it for analysis.  
3. **Model Planning (Phase 3):**  
In this phase, the team determines the methods, techniques, and workflow they intend to follow for the subsequent model building phase.  
This involves choosing appropriate analytical methods (statistical models, machine learning algorithms), defining the variables, and planning the overall approach for model development.  
4. **Model Building (Phase 4):**  
In Phase 4, the team actively develops datasets for testing, training, and production purposes based on the plan from Phase 3.  
In addition to preparing the datasets, the team builds and executes the analytical models based on the work and decisions made in the model planning phase. This involves coding, training models, and initial evaluation.  
5. **Communication Results (Phase 5):**  
In Phase 5, the team collaborates with major stakeholders to determine if the results of the project are a success or a failure, based on the criteria developed in Phase 1.  
This involves presenting findings, insights, and model performance in a clear and understandable manner to the relevant business stakeholders.  
6. **Operationalize (Phase 6):**  
In the final phase, the team delivers the final reports, briefings, code, and technical documents related to the project.  
A key activity in this phase is to put the developed models or analytical results into production. The team may run a pilot project to implement the models in a real-world production environment, ensuring they are integrated into business processes and can deliver value on an ongoing basis.  
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  
Q2)Explain different types of big data analysis techniques. / different kind of big data analysis   
Data analysis techniques vary in their focus, ranging from basic summarization to complex modeling of relationships and predictions. Some basic types include descriptive, exploratory, inferential, predictive, and causal analysis. More specific types may include qualitative analysis (looking for patterns in things like likes, patterns and colors) and quantitative analysis (focuses on numbers), but the primary types explained are:  
1. **Descriptive Analysis:**  
**Question:** Seeks to summarize a characteristic of a set of data. It answers "what happened?"  
**Explanation:** This type of analysis focuses on describing the main features of a dataset. There is no interpretation of the result itself; the result is simply a factual attribute of the data.  
**Example:** Determining the proportion of males in a dataset collected from a group of individuals.

2. **Exploratory Analysis:**  
**Question:** Used to analyze data to see if there are patterns, trends, or relationships between variables. It answers "what patterns or relationships exist?"  
**Explanation:** These types of analyses are also called "hypothesis-generating" analyses. Instead of testing a predefined hypothesis, you are looking for patterns in the data that would suggest a hypothesis.  
**Example:** Looking at sales data over time to see if there's a trend or analyzing customer demographics to see if certain groups tend to buy particular products together.

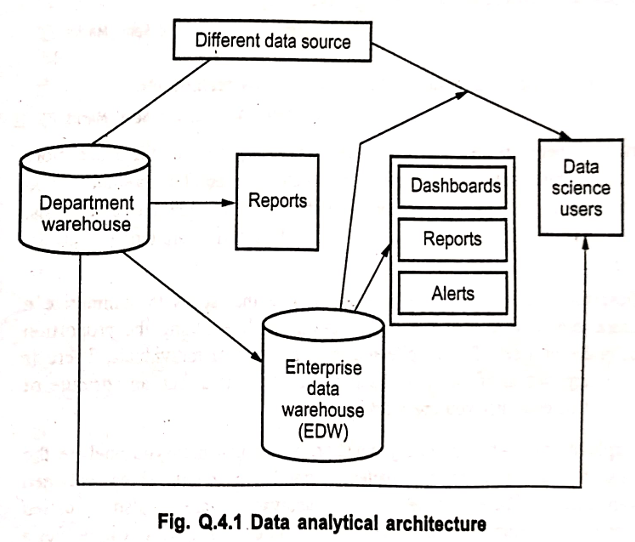
3. **Inferential Analysis:**  
**Question:** A restatement of a proposed hypothesis as a question, answered by analyzing a different set of data. It answers "does the observed pattern hold true in a representative sample or population?"  
**Explanation:** This type of analysis uses data from a sample to make inferences or generalizations about a larger population. It helps determine if an association observed in exploratory analysis holds in a different, representative sample.  
**Example:** After observing a pattern in exploratory analysis that customers in a certain age group buy a specific product, inferential analysis would test if this association is statistically significant and likely holds true for the entire customer base.

4. **Predictive Analysis:**  
**Question:** Asks what types of people or events are likely to occur in the future based on past data. It answers "what might happen?"  
**Explanation:** This analysis uses historical data to forecast future outcomes or probabilities. It's less interested in the cause of an event and more focused on predicting whether the event will happen.  
**Example:** Asking what types of people will eat a diet high in fresh fruits and vegetables during the next year, based on their current habits and demographics. Predicting customer churn or stock prices are other examples.

5. **Causal Analysis:**  
**Question:** Asks about whether changing one factor will change another factor, on average, in a population. It answers "why did it happen?" or "what is the effect of X on Y?"  
**Explanation:** This type of analysis aims to determine the cause-and-effect relationships between variables. It requires careful study design (like A/B testing or controlled experiments) to isolate the impact of one factor on another.  
**Example:** Asking whether changing the price of a product will, on average, change the sales volume.

6. **Mechanistic Analysis:**  
**Question:** Explains how something happens. It answers "how did it happen?" or "what is the mechanism?"  
**Explanation:** This analysis goes beyond simply identifying a causal relationship to understand the underlying process or mechanism by which the effect occurs.  
**Example:** A question that asks how a diet high in fresh fruits and vegetables leads to a reduction in the number of viral illnesses would be a mechanistic question, seeking to understand the biological pathways involved.     
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  
Q3)

Q4) A screenshot of a white sheet

AI-generated content may be incorrect.  
-------------------------------------------**--------------------------------------------------------------------------------------------------------------------------------------  
Q5) big data analytics architecture**  
Analytics architecture refers to the systems, protocols, and technology used to collect, store, and analyze data. It is typically structured in multiple layers, defining how users within an organization can access and interact with data, starting often with a data warehouse architecture.  
"Fig. Q.4.1 Data analytical architecture" illustrates a common representation of such an architecture:  
   
**Different Data Sources:** Data originates from various sources, which can be internal or external to the organization. These sources feed into the analytics ecosystem.  
**Department Warehouse / Local Data Mart:** Data may initially flow into departmental data warehouses or local data marts. These smaller repositories cater to the specific needs and analyses of individual departments or business units.  
**Enterprise Data Warehouse (EDW):** The diagram shows data flowing from departmental warehouses (and potentially directly from sources) into a central Enterprise Data Warehouse (EDW). The EDW is a large, integrated repository designed to store structured and normalized data from across the entire organization.  
**Data Flow Principles:**   
Data to be loaded into the data analytical architecture (like the EDW or warehouses) must be well understood, structured, and normalized according to the appropriate data type.  
Centralization, particularly in the EDW, provides significant benefits, including enhanced security, backup facilities, and the ability to perform pre-processing and establish checkpoints before data is finally stored.  
**Business Intelligence and Reporting Outputs:** From the EDW and potentially departmental warehouses, data is used to generate various outputs for business intelligence and reporting purposes. These include:   
 Reports  
 Dashboards (providing a visual overview of key metrics)  
 Alerts (notifying users of specific conditions or events)  
**Data Science Users and Downstream Analytics:** Data science users interact with the data architecture to perform more in-depth analysis.   
The required level of control on the EDW, combined with additional local systems like departmental warehouses and local data marts, is designed to accommodate business users' need for flexible analysis. Sometimes, these local data marts allow users to conduct some level of more in-depth analysis directly.  
Once data is in the data warehouse, it is also read by additional applications across the enterprise for broader Business Intelligence (BI) and reporting. These are often high-priority operational processes that rely on critical data feeds from the data warehouses and repositories.  
Ultimately, analysts get data provisioned for their downstream analytics. It's noted that sometimes the tools used by analysts are limited to in-memory analytics on desktops, analyzing only samples of data rather than the entire population of a dataset.  
-------------------------------------------**--------------------------------------------------------------------------------------------------------------------------------------  
Q6) short note on CSV and JSON  
CSV (Comma-Separated Values)**A CSV file is a simple text file where data is organized in a structured format, with information separated by commas. CSV files are commonly encountered when working with spreadsheets and databases.Each line in a CSV file typically represents a new record, and the fields (pieces of data within a record) are separated by commas.They are frequently used for transferring data between different storage systems because many applications can easily recognize and process comma-separated records.CSV files can be readily converted to and from various other file formats using applications like Microsoft Excel, which can import data from CSV and save it to formats such as XLS, XLSX, PDF, TXT, XML, and HTML.The CSV file format is specified under RFC4180, which defines its compliance rules, including that each record is generally on a separate line delimited by a line break (CRLF), although the last record may or may not have an ending line break.

**JSON (JavaScript Object Notation)**JSON is a lightweight data-interchange format that is easy for humans to read and write and easy for machines to parse and generate. It is based on a subset of the JavaScript programming language.JSON is primarily used to format data and is commonly used on the Web as a vehicle to describe and transmit data being sent between systems, such as between a server and a web application. It serves as an alternative to XML for this purpose.It is much easier to use with JavaScript compared to XML, which is why JSON Web Services are often replacing XML Web Services in modern web development (especially with Ajax and JavaScript).The JSON format is well-suited for serializing and transmitting structured data over a network connection.JSON's structure is built upon two basic elements: **Objects:** Represented by curly braces {} and containing collections of **property and value** pairs (e.g., "name": "John"). **Arrays:** Represented by square brackets [] and containing ordered lists of **values** (e.g., ["apple", "banana"]).A **value** in JSON can be a string, number, a nested object, an array, a boolean value (true or false), or null.On average, JSON requires fewer characters and thus fewer bytes than the equivalent data represented in XML. Because its syntax is derived from JavaScript, it generally requires less complex parsing when used in applications like Ajax.  
-------------------------------------------**--------------------------------------------------------------------------------------------------------------------------------------  
Q7)** **How data can be ingested in python? Write syntax in python for the same.**Data ingestion is the crucial process of obtaining and importing raw data from various sources into an environment where it can be processed, stored, or immediately used for analysis. This data can originate from diverse sources, including streaming platforms (like Kafka, Kinesis), network connections (TCP sockets), databases, and files in various formats such as Comma Separated Data (CSV), JSON, HTML webpage tables, or Excel.  
In the context of big data analytics in Python, data ingestion typically involves using powerful libraries to read this data from its source and load it into a structured format within Python for subsequent manipulation and analysis. The **Pandas library** is a cornerstone for this, particularly for file-based data, as it facilitates shifting data into the **Pandas DataFrame structure**, which is a highly efficient way to handle structured data.  
The data that is ingested forms a **dataset**. A dataset is fundamentally a collection of records or data points related to a particular subject. In the realm of big data, datasets can broadly be categorized based on their structure:

1. **Tabular Datasets:** These datasets contain structured data organized in a format similar to a table, comprising rows and columns. Each row is a record, and each column represents an attribute. Formats like CSV and Excel inherently represent tabular data.
2. **Non-Tabular Datasets:** These datasets contain unstructured or semi-structured data that doesn't fit neatly into a fixed row-and-column format. Examples include plain text documents, images, audio, video, and semi-structured formats like JSON and XML which have flexible or nested structures.

The method of data ingestion in Python, including the specific syntax used, is often determined by the source of the data and the **type of dataset** being ingested. The goal is to convert the raw data into a suitable Python data structure that accurately represents its original structure (tabular or non-tabular) or transforms it into a more workable format for analysis.  
**Ingesting Data to Form Tabular Datasets in Python:**  
For tabular data sources like CSV or Excel files, Python libraries like Pandas provide straightforward functions to read the data directly into a DataFrame.  
**Method:** Use Pandas functions designed for reading tabular formats. These functions parse the data according to the format's rules (e.g., comma delimiters in CSV) and construct a DataFrame.  
A screenshot of a computer code

AI-generated content may be incorrect.  
**Ingesting Data to Represent Non-Tabular Datasets in Python:**  
Ingesting non-tabular or semi-structured data requires parsing formats that are more flexible than tables. JSON is a common semi-structured format often encountered in web data or APIs. This data can be ingested into Python dictionaries or lists, or sometimes flattened into a DataFrame by Pandas if the structure is consistent.  
**Method:** Use libraries that can parse the specific non-tabular format (like json for JSON, or Pandas read\_json which can handle certain JSON structures).  
**Python Syntax Example (from JSON):**A screenshot of a computer code

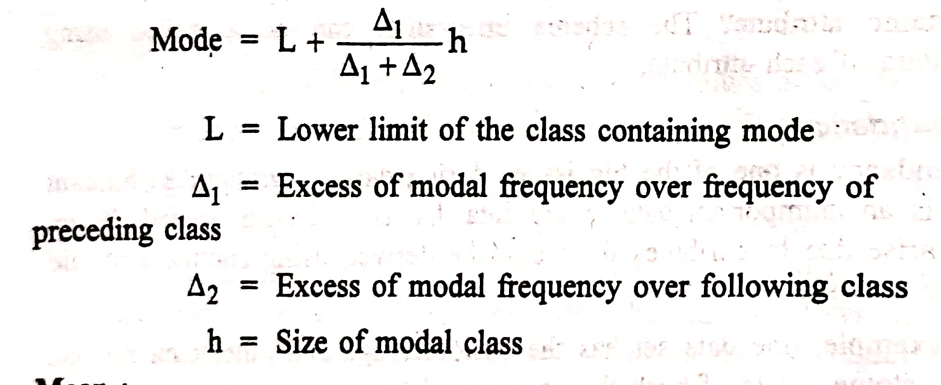
AI-generated content may be incorrect.-------------------------------------------**--------------------------------------------------------------------------------------------------------------------------------------  
q8)Data Cleaning**  
Data preparation is a critical phase in any data analytics or big data project. It involves several tasks to ensure the data is accurate, consistent, and in a usable format. Key aspects of data preparation include data cleaning and data standardization.  
**What is Data Cleaning?**  
Data cleaning is the process of detecting, correcting, or removing corrupt or inaccurate records from a record set, table, or database. It involves identifying incomplete, incorrect, inaccurate, or irrelevant parts of the data and then replacing, modifying, or deleting this "dirty" or "coarse" data. Data cleaning addresses anomalies that potentially exist in all data sources, such as errors and missing values. Failure to clean data properly can lead to numerous problems, including linking errors, model mis-specification, errors in parameter estimation, incorrect analysis, and ultimately, users drawing false conclusions.

The main data cleaning processes include editing, validation, and imputation. Editing identifies errors, validation confirms the correctness of the data and its corrections, and imputation (discussed below) is the replacement of missing values.

**Tasks and Issues Affecting Data Quality:**  
The tasks of data cleaning are typically as follows:  
1. Deal with missing values.  
2. Identify outliers and smooth out noisy data.  
3. Correct inconsistent data.  
These tasks are necessary because several issues can affect the quality of data:  
1. **Invalid values:** Data may contain values that are outside the expected range or format. For instance, a gender field expecting 'F' or 'M' might contain other characters.  
2. **Formats:** A common issue is inconsistent data formats. For example, names might be written as "Name, Surname" or "Surname, Name". Dates, phone numbers, or addresses might also follow different formats.  
3. **Attribute dependencies:** The value of one feature might depend on the value of another in a contradictory way. For instance, if a dataset has a field for "number of students," its value should be related to whether the person is recorded as a "teacher".  
4. **Missing values:** Some features in the dataset may simply have blank or null values.  
5. **Misspellings:** Data may contain incorrectly written values.  
6. **Misfiled values:** Data for one attribute might incorrectly appear in the column intended for another attribute.

**Handling Missing Data:**  
Missing value means data is not always available for certain attributes in records. This can occur due to various reasons: equipment malfunction, inconsistency with other recorded data leading to deletion, data not entered due to misunderstanding, data not considered important at the time of entry, failure to register history or changes, or missing data needing to be inferred because its absence itself carries information.  
Handling missing data is a crucial part of data cleaning. Several methods can be used:  
**Ignore records:** Use only cases (rows) that have all values complete. This is usually done when a critical value like the class label in a prediction task is missing, as many prediction methods do not handle missing class labels well. However, this might not be effective if the percentage of missing values varies considerably across attributes, as it can lead to insufficient and/or biased sample sizes.  
**Ignore attributes:** Remove entire attributes (columns) that have a large percentage of missing values.  
**Use only features (attributes) with all values:** Similar to ignoring attributes, but focusing on keeping only columns that are complete.  
**Fill in the missing value manually:** Replace missing values based on domain knowledge or by manually inspecting a small dataset. This is feasible only for small datasets.  
**Imputation:** Replace missing values with substituted values based on other data in the dataset (e.g., using the mean, median, mode, or more advanced techniques like regression or machine learning models to predict missing values).

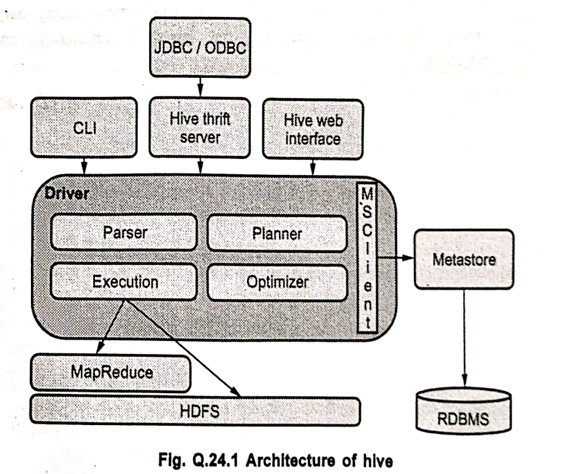
**Data Standardization:**  
Data standardization is a data processing workflow that converts the structure of different datasets into one common format of data. It ensures that data conforms to a standard format that computers can easily read and understand.  
The importance of data standardization lies in:  
**Enabling Data Sharing and Use:** It allows different systems and applications to share and efficiently use data seamlessly. Without standardization, it would be difficult for different approaches to communicate and exchange information effectively.  
**Preserving Data Quality:** Data standardization is essential for preserving data quality. When data is standardized, it becomes much easier to detect errors, inconsistencies, and inaccuracies.  
**Ensuring Accuracy and Reliability:** By making it easier to find and correct data quality issues, standardization helps ensure that the data is accurate and reliable. This is fundamental for making sound decisions based on the data.  
-------------------------------------------**--------------------------------------------------------------------------------------------------------------------------------------  
q9) What is data integration? Discuss issues of data integration.**Data integration is the process of combining data from multiple disparate sources to form a coherent, unified data store. This involves bringing together data that may exist in different formats, structures, and locations into a single, consistent view. Metadata management, correlation analysis, data conflict resolution, and the resolution of semantic heterogeneity (differences in meaning or interpretation of data) all contribute toward achieving smooth data integration.With the increasing volume and velocity of data collected from a variety of sources every day, data integration has become crucially important. Data is often considered the most valuable possession, and integration helps to unlock its full potential.Data integration is important not only because it provides a unified view of scattered data but also because it helps maintain the accuracy of the data by resolving inconsistencies and conflicts between sources.

**Issues of Data Integration:**  
While integrating data from heterogeneous sources, several issues need to be addressed:  
1. **Entity Identification Problem:**  
This problem arises because real-world entities (like a customer) may be represented differently across various data sources.  
The challenge is to "match" these real-world entities from the data originating from different places.  
For example, you might have customer data from two different sources. An entity from one source might have a "customer id," while the same entity in the other source has a "customer\_number." The data analyst or the system needs to understand that these two attributes refer to the same underlying entity.  
Schema integration, which involves combining the schemas (structures) of different data sources, can be achieved using metadata to help resolve this issue.  
2. **Redundancy:**  
Redundancy is one of the big issues during data integration. It refers to data that is either unimportant, no longer needed, or can be derived from other data attributes already present in the integrated dataset.  
For example, if one dataset has the customer's age and another dataset has the customer's date of birth, then age would be a redundant attribute if the date of birth is integrated, as age can be calculated from the date of birth.  
Redundancy can often be discovered using correlation analysis. This involves analyzing the attributes to detect the interdependency or relationship between them, helping to identify which attributes might be redundant.  
-------------------------------------------**--------------------------------------------------------------------------------------------------------------------------------------  
Q10)** **Explain different data transformation techniques.  
I**n data transformation, the data is transformed or consolidated into forms appropriate for mining or analysis. Data transformation is a crucial step in the data preparation phase, ensuring that data is in a suitable format, scale, and structure for the intended analytical tasks.  
Data transformation can involve the following techniques:  
1. **Smoothing:  
Explanation:** This technique helps to remove noise from the data. Smoothing methods include techniques like binning (grouping data into bins or intervals), regression (fitting data to a function), and clustering (grouping similar data points).  
2. **Aggregation:**  
**Explanation:** This involves applying an aggregation or summary operation to the data. Data is summarized to a higher level, which can reduce the data volume and make it more manageable. Examples include summing up sales figures by region or calculating the average temperature per day.  
3. **Generalization:**  
**Explanation:** In generalization, low-level or "primitive" (raw) data are replaced by higher-level concepts. This is achieved through the use of concept hierarchies. For example, specific street addresses might be generalized to cities, states, or regions.  
4. **Normalization:**  
**Explanation:** This technique scales the attribute data so that all values fall within a small, specified range (e.g., between 0 and 1, or between -1 and 1). Normalization is important for algorithms that are sensitive to the scale of the input features, such as many machine learning algorithms.  
5. **Attribute Construction:**  
**Explanation:** This involves constructing and adding new attributes (features) from the given set of existing attributes. These newly constructed attributes can help the mining process by providing more informative features or by combining existing features in a useful way. For example, a "profit margin" attribute could be constructed from "sales revenue" and "cost of goods sold."  
-------------------------------------------**--------------------------------------------------------------------------------------------------------------------------------------  
Q11)** **Explain mean, mode and variance and standard deviation with suitable example.**These are common statistical measures used to describe the central tendency (Mean, Mode) and the dispersion or spread (Variance, Standard Deviation) of a dataset.  
**Mode:** It is the measure of maximum frequency. It occurs most frequently in a dataset. For grouped data, the Mode can be calculated using the formula: A math equations on a white background

AI-generated content may be incorrect.

A math equations and formulas on a white background

AI-generated content may be incorrect.  
  
A math equation with square and square equations

AI-generated content may be incorrect.  
Give your own example   
--------------------------------**-------------------------------------------------------------------------------------------------------------------------------------------------  
Q12) Architecture of Hive**Hive is a data warehousing system built on top of Hadoop. It allows users to access data stored in HDFS and other storage systems using a SQL-like query language called HiveQL. Hive is designed to increase the response time for queries on large datasets, enabling simultaneous access to data and providing a faster response time compared to directly writing complex MapReduce programs for every query.  
The architecture of Hive is depicted in "Fig. Q.24.1 Architecture of Hive" and consists of several major components:  
   
1. **Metastore:**  
The Metastore is the central repository for Hive's metadata.  
This metadata includes information about each table in Hive, such as its schema (column names, data types), its location in HDFS or other storage, and information for partition metadata.  
The Metastore keeps track of the data, replicates it, and provides a backup facility in case of data loss, ensuring the availability of schema information. It typically uses a relational database management system (RDBMS) like MySQL or PostgreSQL to store this metadata persistently.  
2. **Driver:**  
The Driver receives HiveQL statements submitted by various clients.  
It acts like a controller, managing the lifecycle and progress of different query executions through creating sessions  
The Driver stores the metadata generated while executing the HiveQL statement.  
After the execution engine (like MapReduce) completes the processing and the reducing operation is done, the Driver collects the data points and query results and returns them to the client.  
3. **Compiler:**  
The Compiler is tasked with converting a HiveQL query into a form that can be executed by the underlying execution engine (commonly MapReduce, but also Tez or Spark).  
It parses the HiveQL query, performs semantic analysis, and generates an execution plan which is a sequence of steps and tasks needed to let the HiveQL output be generated as needed by the execution engine.  
4. **Optimizer:**  
The Optimizer takes the execution plan generated by the Compiler and applies various transformation steps to improve its efficiency and scalability.  
This can include performing optimizations like predicate pushdown, column pruning, and transforming complex operations (like aggregation and pipeline conversion) potentially by combining multiple joins into a single stage.  
The Optimizer is also assigned to split tasks for improved efficiency before the reduce operations begin.  
5. **Client Interfaces:**  
Hive provides various ways for users and applications to interact with it:  
**CLI (Command Line Interface):** A text-based console for users to enter HiveQL commands.  
**JDBC/ODBC Drivers:** Allow external applications and BI tools to connect to Hive using standard database connectivity protocols.  
**Hive Thrift Server:** Provides a service that allows clients in various programming languages (like Java, Python, C++) to communicate with Hive.  
**Hive Web Interface:** A web-based graphical user interface for interacting with Hive.  
--------------------------------**-------------------------------------------------------------------------------------------------------------------------------------------------  
q13)What is data wrangling? Why do you need it? Explain data wrangling methods?**Data wrangling, also often referred to as data munging, is the iterative process of cleaning, restructuring, transforming, validating, and enriching raw data to make it ready and suitable for analysis and modeling. It involves converting data from its raw format into a more organized and usable structure.

**Why Do You Need It?**  
Data wrangling is a crucial step in almost any data analysis or data science project because real-world data is rarely clean, consistent, or in a format that can be used directly for analysis or feeding into algorithms. Data is often:  
**Messy and inconsistent:** Containing errors, duplicates, or contradictory information.  
**Incomplete:** Having missing values.  
**In incorrect formats:** Data types might be wrong, or different sources might use different conventions.  
**Unstructured or semi-structured:** Not fitting neatly into tables (like text, JSON, XML).  
**Spread across multiple sources:** Requiring combining data from different databases, files, or APIs.

Without proper data wrangling, analyses can be inaccurate, models can perform poorly, and insights derived from the data can be misleading or unreliable. Data wrangling ensures data quality, improves its usability, and prepares it for the analytical tasks ahead, saving significant time and effort in later stages. It is estimated that data professionals spend a large portion of their time (often cited as 60-80%) on data wrangling tasks.

**Explain Data Wrangling Methods:** Data wrangling involves a variety of methods and techniques to prepare data. These methods can be broadly categorized as follows:  
1. **Data Cleaning:**  
**Handling Missing Values:** Identifying and addressing missing data points through methods like imputation (filling in missing values using statistical measures or models), deletion (removing rows or columns with missing values), or flagging them.  
**Dealing with Outliers:** Identifying extreme values that could skew analysis and deciding how to handle them (e.g., removal, transformation, or analysis separately).  
**Correcting Inconsistencies:** Fixing errors in data entry, standardizing inconsistent values (e.g., "USA", "U.S.A.", "United States" all becoming "USA"), and resolving contradictions.

2. **Data Transformation:**  
**Converting Data Types:** Ensuring columns have the correct data types (e.g., converting strings to numbers, dates to datetime objects).  
**Scaling and Normalization:** Adjusting the range of values in numerical columns to a standard scale, important for many algorithms (e.g., min-max scaling, z-score normalization).  
**Creating New Features (Attribute Construction):** Deriving new, potentially more informative features from existing ones (e.g., calculating BMI from height and weight, extracting month or year from a date).  
**Aggregating Data:** Summarizing data by grouping it based on certain attributes (e.g., calculating total sales per month, average rating per product).

3. **Data Structuring and Restructuring:**  
**Parsing Data:** Extracting data from unstructured or semi-structured formats (like parsing text, JSON, XML) into a structured format.  
**Reshaping Data:** Changing the layout of the data, such as pivoting (transforming rows into columns) or melting (transforming columns into rows) to make it suitable for analysis.  
**Combining Data:** Merging or joining data from multiple tables or sources based on common keys or relationships.

4. **Data Enrichment:**   
Adding valuable information to the existing dataset from external sources (e.g., adding demographic data based on zip codes, adding weather data based on location and time).

5. **Data Validation:**  
Implementing checks to verify the quality and consistency of the data against predefined rules or constraints after wrangling steps have been performed.

**Example Scenario:** Suppose you're analyzing customer data from two sources. One file has customer names and emails, the other has their purchase history. Before analysis, you would:Clean missing emailsMerge both datasets using Customer ID  
Encode purchase categoriesRemove outliers in purchase amountConvert date fields into a standard format  
--------------------------------**-------------------------------------------------------------------------------------------------------------------------------------------------  
q14)** **What is categorical variable? Why do you need categorical variable encoding? With an example, explain one-hot encoding?  
What is a Categorical Variable?**  
A categorical variable is a variable that can take on one of a limited, and usually fixed, number of possible values. These values represent categories or labels. Instead of having numerical values that can be measured on a scale, a categorical variable places an observation into a particular group or category.     
**Examples:** Gender (Male, Female, Other), Color (Red, Blue, Green), City (New York, London, Tokyo), Education Level (High School, Bachelor's, Master's, PhD), Yes/No responses.

Categorical variables can be:  
**Nominal:** Categories do not have a natural order or ranking (e.g., Color, City, Gender).  
**Ordinal:** Categories have a clear order or ranking (e.g., Education Level, Customer Satisfaction (Low, Medium, High)).

**Why Do You Need Categorical Variable Encoding?**  
You need categorical variable encoding because most machine learning algorithms are designed to operate on numerical data. They use mathematical equations and calculations that require numerical inputs. Categorical variables, represented by text labels or distinct categories, cannot be directly processed by these algorithms.  
Encoding converts these categorical variables into a numerical format that machine learning models can understand and work with. This transformation is essential to use categorical data effectively in building predictive models.

**Explain One-Hot Encoding with an Example:** One-hot encoding is one of the most common techniques for encoding categorical variables, particularly nominal ones where there is no inherent order between categories.  
**Explanation:** One-hot encoding transforms a categorical variable with 'N' unique categories into 'N' new binary variables (columns). For each observation (row), only one of these new columns will have a value of 1 (indicating that the observation belongs to that category), while all other 'N-1' columns for that variable will have a value of 0.  
This method avoids creating an artificial sense of order or numerical relationship between categories that might be implied by other simple numerical encoding methods (like assigning integers 1, 2, 3, etc., which could mislead an algorithm into thinking 3 is "greater than" 1).

**Example of One-Hot Encoding:** Let's say we have a dataset with a categorical variable 'Color':  
**Original Data:**   
**A screenshot of a phone

AI-generated content may be incorrect.**

The 'Color' variable has three unique categories: Red, Blue, and Green. Using one-hot encoding, we will replace the single 'Color' column with three new columns, one for each unique color category.

**Data after One-Hot Encoding:**A screenshot of a computer

AI-generated content may be incorrect.  
As you can see, each unique color value from the original 'Color' column is now represented by a '1' in its corresponding new binary column and '0's in the others. This allows machine learning algorithms to process the categorical information numerically without assuming any ranking between 'Red', 'Blue', or 'Green'.  
--------------------------------**-------------------------------------------------------------------------------------------------------------------------------------------------**  
**q15)**A screenshot of a computer

AI-generated content may be incorrect.--------------------------------**-------------------------------------------------------------------------------------------------------------------------------------------------  
q16) How missing values are filled in Pandas Data Frame with zeros? Assume suitable data.**In the Pandas library in Python, missing values are typically represented by NaN (Not a Number). You can fill these missing values with zeros using the .fillna() method of a DataFrame or Series.

Let's assume a suitable DataFrame with some missing values:  
A screenshot of a computer code

AI-generated content may be incorrect.  
**Explanation:**  
1. We import the pandas library and numpy (often used with pandas, especially for np.nan).  
2. We create a sample DataFrame df with some NaN values in columns 'A' and 'B' to represent missing data.  
3. The df.fillna(0) method is called on the DataFrame. This method returns a *new* DataFrame where all occurrences of NaN have been replaced by 0.  
4. The original DataFrame df remains unchanged unless you assign the result back to it (df = df.fillna(0)) or use the inplace=True argument (df.fillna(0, inplace=True) - though inplace=True is often discouraged in favor of explicit assignment for clarity).  
--------------------------------**-------------------------------------------------------------------------------------------------------------------------------------------------  
q17)** **Explain Min-max scaling. For the following dataset carry out min-max Scaling, X = 24,28,53,30,40,18,15,21.**Min-max scaling (also known as normalization) is a data transformation technique that rescales the range of features to lie within a fixed range, typically between 0 and 1. This scaling is often used in machine learning algorithms (like support vector machines, neural networks) that are sensitive to the magnitude or scale of input features. **A math equation on a white background

AI-generated content may be incorrect.A math equation with numbers

AI-generated content may be incorrect.  
A screenshot of a math test

AI-generated content may be incorrect.A number of numbers on a white background

AI-generated content may be incorrect.**