multiverse

Data Analysis Tools

Revision Materials Pack

Welcome to your Data Fellowship BCS Data Analysis Tools Revision Pack.

This Revision Pack has been created by Multiverse to help you prepare for your BCS Data Analysis Tools exam - the second exam you'll take as part of your data fellowship.

We have intentionally left lots of extra space to allow you to add your own annotations and notes.

The format for the examination is a one-hour multiple-choice examination consisting of 40 questions. The examination is closed book (no materials can be taken into the examination room). The pass mark is 26/40 (65%).

Explain the purpose and outputs of data integration activities

Integration begins with the ingestion process, and includes steps such as cleansing, ETL mapping, and transformation. Data integration ultimately enables analytics tools to produce effective, actionable business intelligence.

Functional Vs Non-functional requirements

A **functional** requirement defines a system or its component. It describes the functions a software must perform. A function is nothing but inputs, its behaviour, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.
Functional software requirements help you to capture the intended behavior of the system. This behavior may be expressed as functions, services or tasks or which system is required to perform.
A **non-functional** requirement defines the quality attribute of a software system. They represent a set of standards used to judge the specific operation of a system. Example, how fast does the website load?
A non-functional requirement is essential to ensure the usability and effectiveness of the entire software system. Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.
Non-functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users is > 10000. Description of non-functional requirements is just as critical as a functional requirement.

Speed and time available

Speed of data integration output is faster for data warehouses than relational databases because the access is only write-only. Data warehouses allow large analytical queries which eliminate the issue by accessing transactional databases (OLTP).
Loading data into warehouses (ETL) is usually carried out in batches, (every few days or every week etc) and during the loading stage, the data warehouse is unavailable, which is why data is loaded in batches.
However, having data integrated into a single source saves time; as it doesn't take as long to prepare and analyse the data.

Information structure and rules

Data integration activities for data warehouses requires that you follow some basic rules:

- ➔ Security policies must be specified by organisations providing data sources with relations to processing the data. This is to prevent data leakage and unauthorized access.
- ➔ **Access layers** (including networks, servers, firewalls) between the data source and targets should be properly configured especially if data is sourced externally.
- ➔ Data integrated should be immutable. You should not be able to change the content of the integrated data destination. In other words, it will only allow reading output and the only input should be loading data from the different sources.
- ➔ **Validation checks** should be carried out during ETL.
- ➔ Validate the **source and target table structure, data types.**
- ➔ Validate the **column names against a mapping document**.
- ➔ Verification is also carried out on the Data Warehouse. This is done using ETL testing
- ➔ Verify that the data is accurate
- ➔ Verify the data is the right data required to be in the data warehouse.
- ➔ Verify that data has not duplicated in the data warehouse

Rationale for using and integrating data from multiple sources

Using and integrating data from multiple sources is extremely important. Imagine a scenario where the data needed is spread out over many divisions in a company or the different departments store their data in different ways.

Ensuring **consistency** across the firm, spotting missing data and identifying gaps that need to be filled is achieved through ETL processes. For example, we may want to track an order from sales, through inventory and finally to shipping. ETL allows us to combine these different data sets and find insights.

Other benefits of integrating data from multiple sources within an organization include **increased collaborations** across teams, **better business intelligence and insight** for making business decisions and **data availability** to all stakeholders.

Importance of data in a business context

In its simplest form, data is essentially the plain facts and statistics collected during the operations of a business. It's as simple as that. Its importance, however, can vary widely depending on who is using it and in what context. While the data itself may not be very informative on its own, it is the basis for all reporting and as such is crucial in business.

Data is no longer restricted to just technological companies. Businesses as diverse as life-insurance agencies, hotels, and product management companies are now using data to improve their marketing strategies, customer experience, and to understand business trends or just collect insights on user data.

Companies use data for a myriad of purposes, depending on their industry, but there are usually three key aspects we are looking at:

1. *Variability:* Illustrates how things differ, and by how much
2. *Uncertainty:* Good visualization practices frame uncertainty that arises from variation in data
3. *Context:* Meaningful context helps us frame uncertainty against underlying variation in data

Note Space

Explain how data from multiple sources and systems can be integrated to provide a unified view of the data

Business need for analysis;

Data analytics is used in business to help organisations make better business decisions.
Whether it's market research, product research, positioning, customer reviews, sentiment analysis, or any other issue for which data exists, analyzing data will provide insights that organisations need in order to make the right choices.
Data analytics is important for businesses today, because data-driven choices are the only way to be truly confident in business decisions. Some successful businesses may be created on a hunch, but almost all successful business choices are data-based.
Data is not a universal ubiquitous solution to all of a business' problems. Without analysis, data has limited usage. While data can produce a lot of valuable insights after analysis, data does not contain all of the answers – it can only display the best picture of the data based on the data itself and the analysis an analyst decides to do.
Data analysis is a somewhat abstract concept to understand without the help of examples. So to better illustrate how and why data analysis is important for businesses, here are 4 types of data analysis and examples of each.

- **Descriptive Analysis:** Descriptive data analysis looks at past data and tells what happened. This is often used when tracking Key Performance Indicators (KPIs), revenue, sales leads, and more.
- **Diagnostic Analysis:** Diagnostic data analysis aims to determine why something happened. Once your descriptive analysis shows that something negative or positive happened, diagnostic analysis can be done to figure out the reason. A business may see that leads increased in the month of October and use diagnostic analysis to determine which marketing efforts contributed the most.
- **Predictive Analysis:** Predictive data analysis predicts what is likely to happen in the future. In this type of research, trends are derived from past data which are then used to form predictions about the future. For example, to predict next year's revenue, data from previous years will be analyzed. If revenue has gone up 20% every year for many years, we would predict that revenue next year will be 20% higher than this year. This is a simple example, but predictive analysis can be applied to much more complicated issues such as risk assessment, sales forecasting, or qualifying leads.
- **Prescriptive Analysis:** Prescriptive data analysis combines the information found from the previous 3 types of data analysis and forms a plan of action for the organisation to face the issue or decision. This is where the data-driven choices are made.

Reasons for using data from multiple sources;

Data within an enterprise originates from many different sources, for example we can have advertising data, inventory data, sales data, customer support data etc. Having access to multiple data sources allows the analyst to pull together data to resolve issues, perform data analysis, obtain a 360 view of a problem/situation or assist with decision making among others.
An industry standard tool that allows the collection and organisation of data originating from various sources, is the *data warehouse.*

Importance of data source quality to improve the quality of results;

The quality of a data analysis report depends inherently on the quality of the data source. When dealing with data, many things could go wrong, including:

- Data Truncation – e.g. precision is lost, wrong data types.
- Data Corruption – e.g. commas in the wrong place.
- Data Missing – e.g. only a portion of the data set is uploaded (missing rows).
- Data Typing – e.g. wrong data type uploaded into field / data mismatching.
- Data Translation – e.g. wrong encoding, wrong/symbols and rich characters lost.

These kinds of data errors may lead to wrong business decisions that could prove catastrophic for the enterprise. Some suggestions to ensure better quality of data include the following:

- ➔ CheckSum
- ➔ Spot Checks (Eyeballing)
- ➔ Mins/Max/Aggregates
- ➔ Counts
- ➔ Export Comparison

Filtering data to ensure only relevant data is combined to underpin business objectives

In order to improve our decision making we need to gather data. This data could come from multiple sources (files, spreadsheets, databases, applications).
These different sources - the field naming conventions, file layouts, character encoding, etc. - might be different in each source.
Every organisation should have a data integration strategy as the volumes and complexity of the data they are required to manage escalates rapidly. Without a data integration strategy the natural complexity of interfaces between applications quickly becomes unmanageable.
A data integration strategy may be part of the data management strategy or simply part of the technology strategy, but it should include batch, real-time, and big data integration management.

What is data integration?

Data integration is the process of combining data from different sources to help data managers and executives analyse it and make smarter business decisions.
This process involves a person or system locating, retrieving, cleaning, and presenting the data.
Data managers and/or analysts can run queries against this merged data to discover business intelligence insights.
With so many potential benefits, businesses need to take the time to align their goals with the right approach.
Data integration is the key to how we will get good quality information, we can look at a data set by itself, but we could enrich the analysis by combining multiple data sets that may not all be stored in the same place.
We can have transactional information, but we can combine this with other interesting data to understand what trends we are seeing.
For example, Tesco will combine their sales data for products with weather data to see if there is a correlation between the temperature and the sales figures. This is how they can forecast how much lettuce to buy from suppliers in the hotter months for BBQs.

Data integration techniques

There are five types of data integration we typically come across in our day-to-day roles. These are sometimes referred to as approaches or techniques (source: Talend).

1. Manual data integration

Manual data integration occurs when a data manager oversees all aspects of the integration — usually by writing custom code. That means connecting the different data sources, collecting the data, and cleaning it, etc., without automation.
Some of the benefits are:

- ➔ **Reduced cost:** This technique requires little maintenance and typically only integrates a small number of data sources.
- ➔ **Greater freedom:** The user has total control over the integration.

Some of the cons are:

- ➔ **Less access:** A developer or manager must manually orchestrate each integration.
- ➔ **Difficulty scaling:** Scaling for larger projects requires manually changing the code for each integration, and that takes time.
- ➔ **Greater room for error:** A manager and/or analyst must handle the data at each stage.

This strategy is best for one-time instances, but it quickly becomes untenable for complex or recurring integrations because it is a very tedious, manual process. Everything from data collection, to cleaning, to presentation is done by hand, and those processes take time and resources.

2. Middleware data integration

Middleware is software that connects applications and transfers data between them and databases. It's especially handy when a business is integrating stubborn legacy systems with newer ones, as middleware can act as an interpreter between these systems.

Some of the benefits are:

- ➔ **Better data streaming:** The software conducts the integration automatically and in the same way each time.
- ➔ **Easier access between systems:** The software is coded to facilitate communication between the systems in a network.

Some of the cons are:

- ➔ **Less access:** The middleware needs to be deployed and maintained by a developer with technical knowledge.
- ➔ **Limited functionality:** Middleware can only work with certain systems.

For businesses integrating legacy systems with more modern systems, middleware is ideal, but it's mostly a communications tool and has limited capabilities for data analytics.

3. Application-based integration

In this approach, software applications do all the work. They locate, retrieve, clean, and integrate data from disparate sources. This compatibility makes it easy for data to move from one source to the other.

Some of the benefits include:

- ➔ **Simplified processes:** One application does all the work automatically.
- ➔ **Easier information exchange:** The application allows systems and departments to transfer information seamlessly.
- ➔ **Fewer resources are used:** Because much of the process is automated, managers and/or analysts can pursue other projects.

Some of the cons include:

- ➔ **Limited access:** This technique requires special, technical knowledge and a data manager and/or analyst to oversee application deployment and maintenance.
- ➔ **Inconsistent results:** The approach is unstandardized and varies from businesses offering this as a service.
- ➔ **Complicated setup:** Designing the application(s) to work seamlessly across departments requires developers, managers, and/or analysts with technical knowledge.
- ➔ **Difficult data management:** Accessing different systems can lead to compromised data integrity.

Sometimes this approach is called enterprise application integration, because it's common in enterprises working in hybrid cloud environments. These businesses need to work with multiple data sources — on-premises and in the cloud. This approach optimizes data and workflows between these environments.

4. Uniform access integration

This technique accesses data from even more disparate sets and presents it uniformly. It does this while allowing the data to stay in its original location.

Some of the advantages are:

- ➔ **Lower storage requirements:** There is no need to create a separate place to store data.
- ➔ **Easier data access:** This approach works well with multiple systems and data sources.
- ➔ **Simplified view of data:** This technique creates a uniform appearance of data for the end user.

Some of the difficulties are:

- ➔ **Data integrity challenges:** Accessing so many sources can lead to compromising data integrity.
- ➔ **Strained systems:** Data host systems are not usually designed to handle the amount and frequency of data requests in this process.

For businesses needing to access multiple, disparate systems, this is an optimal approach. If the data request isn't too burdensome for the host system, this approach can yield insights without the cost of creating a backup or copy of the data.

5. Common storage integration (sometimes referred to as data warehousing)

This approach is similar to uniform access, except it involves creating and storing a copy of the data in a data warehouse. This leads to more versatility in the ways businesses can manipulate data, making it one of the most popular forms of data integration.

Some of the benefits include:

- ➔ **Reduced burden:** The host system isn't constantly handling data queries.
- ➔ **Increased data version management control:** Accessing data from one source, versus multiple disparate sources, leads to better data integrity.
- ➔ **Cleaner data appearance:** The stored copy of data allows managers and/or analysts to run numerous queries while maintaining uniformity in the data's appearance.

➔ **Enhanced data analytics:** Maintaining a stored copy allows manager and/or analysts to run more sophisticated queries without worrying about compromised data integrity.

Some of the cons include:

➔ **Increased storage costs:** Creating a copy of the data means finding and paying for a place to store it.

➔ **Higher maintenance costs:** Orchestrating this approach requires technical experts to set up the integration, oversee, and maintain it.

Common storage is the most sophisticated integration approach. If businesses have the resources, this is almost certainly the best approach, because it allows for the most sophisticated queries. That sophistication can lead to deeper insights.

Here's an overview of the best scenario for each approach:

Data integration approach	When to use it
Manual data integration	Merge data for basic analysys between a small amount of data sources
Middleware data integration	Automate and translate communication between legacy and modernised systems
Application-based integration	Automate and translate communication between systems and allow for more complicated data analysis
Uniform access integration	Automate and translate communication between systems and present the data uniformly to allow for complicated data analysis
Common storage integration	Present the data uniformly, create and store a copy, and perform the most sophisticated data analysis tasks

Common user interface

With Manual Integration or **Common User Interface,** users operate with all the relevant information accessing all the source systems or web page interface. No unified view of the data exists.

Data visualisation can be displayed across a series of multiple charts, in UIs called **dashboards**. Multiple, separate charts can sometimes better communicate a story, rather than one complex chart.

A **dashboard** informs users what they are doing, a **scorecard** tells them how well they are doing. In other words, a **dashboard** records performance while a **scorecard** charts progress. A **dashboard** is typically used as a performance monitoring system, whereas a **scorecard** is a performance management system.

These dashboards typically include a few small charts or a scorecard, with **dynamic** headlines that explain the trends and insights provided in each supporting chart.

Use cases include:

➔ Providing an overview of key performance indicators

➔ Creating a high-level executive summary

Virtual Integration

Uniform Data Access or **Virtual Integration** - leaves data in the source systems and defines a set of views to provide and access the unified view to the customer across the whole enterprise.

For example, when a user accesses the customer information, the particular details of the customer are transparently acquired from the respective system.

The main benefits of the virtual integration are nearly zero latency of the data updates propagation from the source system to the consolidated view, no need for a separate store for the consolidated data.

However, the drawbacks include limited possibility of data's history and version management, limitation to apply the method only to 'similar' data sources (e.g. same type of database) and the fact that the access to the user data generates extra load on the source systems which may not have been designed to accommodate users.

Physical data integration

Physical data integration can be achieved through an ETL process. Each stage involved in an ETL process is outlined in the table below.

Extract	Transform	Load
Extraction is the process of reading multiple data sources into the Data Warehouse, where data is COPIED not moved. Data Validation occurs during this stage, you must have the correct: ➔ Structure ➔ Format ➔ Permissions Method of extraction—for each data source, define whether the extraction process is manual or tool-based.	Transformation is the process of combining the data tables or linking them using relational databases. The data itself is not changed in any way. Data engineers must consider the efficiency of the databases as well as ensuring that all necessary data can be accessed. Before moving the data into a single point of reference we need to: ➔ Remove inconsistencies ➔ Standardise various data elements such as naming conventions and codes ➔ Make sure of the meanings of the data names in each file / database / application ➔ Deduplication ➔ Deriving new calculated values ➔ Data Validation	Loading is the process of writing the data to the target database. Due to the nature of Big Data it has become necessary to use parallel processing to manage the volume of data being written to the system. Data Verification is undertaken post-loading to ensure the data is accurate.

Note Space

Describe how programming languages for statistical computing (SQL) can be applied to data integration activities, improving speed and data quality for analysis.

Programming constructs

Programming constructs are essential components of any program. ***Sequence***, ***Selection***, and ***Iteration*** are the basic elements that we use to tell the computer what to do. The code will definitely look different depending on the programming language we use, but the algorithm will be the same.
The role of these components is described in the table below:

Sequence	Selection	Iteration
It is the order we want the computer to execute the instructions we provide as programmers. For example, do this first, then do this, then do that, and so forth.	Selecting which path of an algorithm to execute depending on some criteria. For example, if you passed a class in school, then we execute the operations that clap and cheer and play a song. But if you didn't pass the class, then maybe we would say, "Better luck next time, hang in there!"	Refers to looping or repeating procedures. Many times, we want to be able to repeat a set of operations a specific number of times or until some condition occurs..

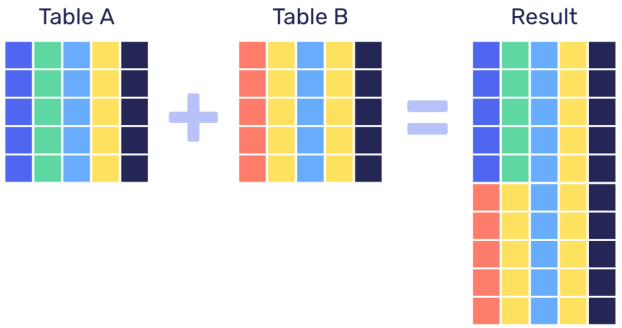
Single queries

`SELECT` is the foundation of almost every query we write - it tells the database which columns we want to see. We can either specify columns by name (separated by commas) or use the wildcard `*` to return every column in the table.

The SQL SELECT general form	Example
SELECT: column-names	SELECT: FirstName, LastName, City, Country
FROM: table-name	FROM: Customer
WHERE: condition	WHERE: City = **"Paris"**
ORDER BY: sort-order	ORDER BY: LastName

Multiple queries (UNION); Joins with duplicate values

UNION is another way to return information from multiple tables with a single query. The UNION statement allows you to perform queries against several tables and return the results in a consolidated set, as in the following example:



A **UNION** merges rows of similar data to create a new set

Four basic rules for using UNION are the following:

1. You must match the number of columns and they must be of compatible data types.
2. You can only have one ORDER BY at the bottom of your full SELECT statement.
3. UNION removes exact duplicates; UNION ALL allows duplicates.
4. Conditions between UNION SELECT statements should match.

Expressions

A SQL expression is a combination of one or more values, operators and SQL functions that results in a value. These SQL EXPRESSIONs are similar to a formula and they are written in query language. You can also use them to query the database for a specific set of data. Some important expressions/statements you need to remember are:

- ➔ CASE: groups data into **categories** or **classifications**. In SQL, CASE is part of the columns list in the SELECT statement. The CASE statement goes through conditions and returns a value when the first condition is met (like an IF-THEN-ELSE statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause. If there is no ELSE part and no conditions are true, it returns NULL.
- ➔ DATETIME: type of variable that allows storing a date/time value in the database, in the format: YYYY-MM-DD HH:MI:SS
- ➔ Compound: refers to the mathematical operations within the relational databases. Operators are divided into four categories: *Arithmetic, Comparison, Logical, String*. Examples of compound operators include the following:

Operator Type	Characters	Description
Arithmetic	+ - * / **	Addition or prefix plus. Subtraction or prefix minus. Multiplication. Division. Exponent (to the power of).
Comparison	= != or <> < <= or !> > >= or !<	Equal to. Not equal to. Less than Less than or equal to (not greater than). Greater than. Greater than or equal to (not less than).
Logical	ALL / AND ANY / OR BETWEEN EXISTS IN LIKE NOT IS NULL UNIQUE	Does the value meet **ALL** criteria in list? Does the value meet **ANY** criteria in list? Is the value **BETWEEN** listed values? Does a row meeting the clitexia **EXIST** in the data? Is the value found in the listed **literal** values? **Compares** the value to listed values using wildcards. **Reverses** the meaning of a logical operator. Checks if the value is **NULL.** Searches all rows for **duplicates.**
String	CHAR(n) CONCAT() FORMAT(n) LOWER() UPPER() REPEAT() TRIM()	Returns the character at index **n**. **Concatenates** (puts together) items. Returns a number formatted to **n** decimal places. Returns the argument in **lowercase**. Returns the argument in **uppercase**. **Repeats** the string a certain number of times. **Removes** leading and trailing spaces.

Functions

SQL has many built-in functions for performing calculations on data.
SQL aggregate functions return a single value, calculated from values in a column.

- ➔ AVG returns the average value
- ➔ COUNT returns the number of rows
- ➔ MAX returns the largest value
- ➔ MIN returns the smallest value
- ➔ GROUPBY indicates the dimensions you want to group your data by (e.g. a category that you wish to sort into subgroups).
- ➔ ROUND rounds a number to a specified number of decimal places
- ➔ CAST converts an expression from one datatype to another datatype. If the conversion fails, the function will return an error. Otherwise, it will return the converted value. In many cases, fields need to be CAST before an aggregation can be performed.
- ➔ CONVERT converts a value (of any type) into a specified datatype.
- ➔ ISNULL returns a specified value if the expression is NULL. If the expression is NOT NULL, this function returns the expression.

Querying multiple tables in different information

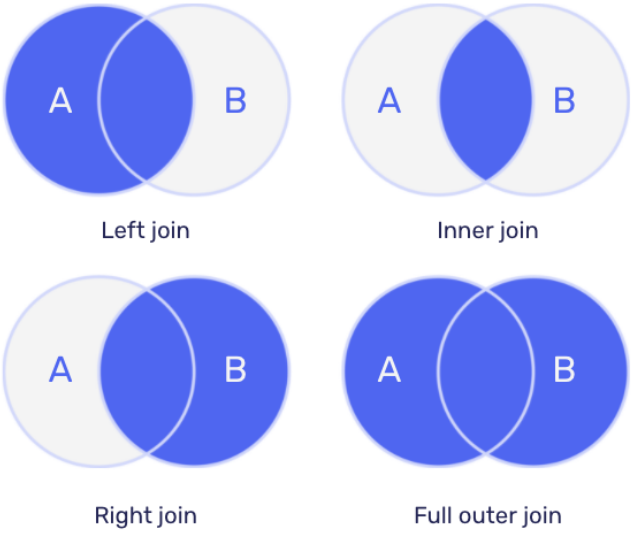
The JOIN statement in SQL is similar to EXCEL's VLOOKUP. Both commands, connect data sources together in order to use information from both tables to display a desired result.
When you create a JOIN, each table can have an ALIAS and each COLUMN is connected to the TABLE by the ALIAS. An ALIAS is a shorthand name given to tables or columns in a table that you intend to reference repeatedly. Types of JOIN include the following:

- ➔ INNER: The inner join clause links two (or more) tables by a relationship between two columns. Whenever you use the inner join clause, you normally think about the intersection.
- ➔ OUTER: a full outer join is the combination of a left join and a right join.
- ➔ FULL(OUTER): includes all rows from the joined tables whether or not the other table has the matching row.returns the number of rows
- ➔ RIGHT: returns all records from the right table (table2), and the matched records from the left table
- ➔ LEFT: returns all rows from the left table whether or not there is a matching row in the right table.
- ➔ UNION: combines result sets of two or more SELECT statements into a single result set.
- ➔ SELECT INTO: copies data from one table into a new table.
- ➔ SUBQUERIES: queries within another SQL query and embedded within the WHERE clause. A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved

Joining on multiple fields

You can create relationships on multiple tables using a range of JOINS:

- ➔ INNER, RIGHT, and LEFT JOINS
- ➔ FULL OUTER JOINS
- ➔ EXCEPTION JOINS
  - OUTER and EXCEPTION JOINs assist you in handling missing data between tables.
- ➔ CROSS JOINS.
  - A CROSS JOIN matches every row of the primary table with every row of the secondary table. This type of JOIN results in a Cartesian product of the tables. This is generally detrimental to fast performance and not desired.



Select and select* statements

SQL is used to communicate questions to the database. Few of the main clauses are the following:

- ➔ SELECT
  - Allows you to select certain columns from a table.
  - Determines which columns of information are downloaded.
- ➔ SELECT*
  - Allows you to select all columns from a table.
- ➔ FROM
  - Specifies the tables from which the query extracts data.
  - Defines the relationships between the tables (JOIN conditions).
- ➔ WHERE
  - Filters which rows are selected from the tables.
    - AND
    - Returns TRUE if both conditions are true.
    - OR
    - Returns FALSE if neither condition is true (TRUE if either is true).
    - ORDER BY
    - Sorts the results
    - % or - (Wildcard)
    - substitutes one (-) or more (%) characters in a string. It is used when you want to access the data in a column that start, contain or end with specific character(s).

Selecting the first / last of occurrences

ORDER BY sorts results in an ascending or descending order. This enables you to find and select the first/last of occurrences in a column.
After the ORDER BY is a number that indicates the column by which you're sorting.
The default sort order is ascending, but you can specify ascending (ASC) or descending (DESC) to determine the sort order.
To obtain one item, you should limit the selection to 1.
Examples (MySQL):

➔ SELECT * FROM Table ORDER BY ID Key DESC LIMIT 1

➔ SELECT * FROM Table ORDER BY ID Key ASC LIMIT 1

Implicit data conversion

Implicit data conversion is when the SQL server automatically converts the data from one type to another type during the query execution process. It is called an 'implicit' process because it happens behind the scenes, automatically without the user having any power over it.
The mismatched data types have to be converted to compatible formats by the SQL Server and this data type conversion is done according to a defined process governed by the SQL precedence. Find out more about implicit data conversion, here.

Note Space

Explain how to take account of data quality in preparing data for analysis to improve accuracy, quality and usefulness.

Data Profiling

Data profiling is a process of examining data from an existing source and summarising information about that data. You profile data to determine the accuracy, completeness, and validity of your data. Data profiling can be done for many reasons, but it is most commonly part of helping to determine data quality as a component of a larger project.

Commonly, data profiling is combined with the ETL process to move data from one system to another. When done properly, ETL and data profiling can be combined to cleanse, enrich, and move quality data to a target location.

For example, you might want to perform data profiling when migrating from a legacy system to a new system. Data profiling can help identify data quality issues that need to be handled in the code when you move data into your new system.

Or, you might want to perform data profiling as you move data to a data warehouse for business analytics. Often when data is moved to a data warehouse, ETL tools are used to move the data. Data profiling can be helpful in identifying what data quality issues must be fixed in the source, and what data quality issues can be fixed during the ETL process.

Data profiling can be performed in different ways, but there are roughly three base methods used to analyse the data.

Column profiling counts the number of times every value appears within each column in a table. This method helps to uncover the patterns within your data.

Cross-column profiling looks across columns to perform key and dependency analysis. Key analysis scans collections of values in a table to locate a potential primary key. Dependency analysis determines the dependent relationships within a data set. Together, these analyses determine the relationships and dependencies within a table.

Cross-table profiling looks across tables to identify potential foreign keys. It also attempts to determine the similarities and differences in syntax and data types between tables to determine which data might be redundant and which could be mapped together.

Rule validation is sometimes considered the final step in data profiling. This is a proactive step of adding rules that check for the correctness and integrity of the data that is entered into the system.

What happens when an error or issue is found

In general, incorrect data should be either **rejected**, **corrected**, or **imputed**.

Reject
If the missing values in a column rarely happen and occur at random, then the easiest and most forward solution is to drop observations (rows) that have missing values.
If most of the column's values are missing, and occur at random, then a typical decision is to drop the whole column.
This is particularly useful when doing statistical analysis, since filling in the missing values may yield unexpected or biased results.
Impute
It means to calculate the missing value based on other observations. There are quite a lot of methods to do that.

First. One is using statistical values like mean, median. However, none of these guarantees unbiased data, especially if there are many missing values.

Mean is most useful when the original data is not skewed, while the median is more robust, not sensitive to outliers, and thus used when data is skewed.

In a normally distributed data, one can get all the values that are within 2 standard deviations from the mean. Next, fill in the missing values by generating random numbers between (mean — 2 * std) & (mean + 2 * std)

```
rand = np.random.randint(average_age - 2*std_age, average_age + 2*std_age, size = count_nan_age)
dataframe["age"][np.isnan(dataframe["age"])] = rand
```

Second. Using a linear regression. Based on the existing data, one can calculate the best fit line between two variables, say, house price vs. size m².
It is worth mentioning that linear regression models are sensitive to outliers.
Third. Hot-deck: Copying values from other similar records. This is only useful if you have enough available data. And, it can be applied to numerical and categorical data.
One can take the random approach where we fill in the missing value with a random value. Taking this approach one step further, one can first divide the dataset into two groups (strata), based on some characteristic, say gender, and then fill in the missing values for different genders separately, at random.
In sequential hot-deck imputation, the column containing missing values is sorted according to auxiliary variable(s) so that records that have similar auxiliaries occur sequentially. Next, each missing value is filled in with the value of the first following available record.
What is more interesting is that k nearest neighbour imputation, which classifies similar records and put them together, can also be utilized. A missing value is then filled out by finding first the k records closest to the record with missing values.
Next, a value is chosen from (or computed out of) the k nearest neighbours. In the case of computing, statistical methods like mean (as discussed before) can be used.
Correct
Some argue that filling in the missing values leads to a loss in information, no matter what imputation method we used.
That's because saying that the data is missing is informative in itself, and the algorithm should know about it. Otherwise, we're just reinforcing the pattern already existing by other features.
This is particularly important when the missing data doesn't happen at random. Take for example a conducted survey where most people from a specific race refuse to answer a certain question.
Missing numeric data can be filled in with say, 0, but these zeros must be ignored when calculating any statistical value or plotting the distribution.
While categorical data can be filled in with say, "Missing": A new category which tells that this piece of data is missing.
Remember
Missing values are not the same as default values. For instance, zero can be interpreted as either missing or default, but not both.
Missing values are not "unknown". A conducted research where some people didn't remember whether they have been bullied or not at the school, should be treated and labelled as unknown and not missing.
Every time we drop or impute values we are losing information. So, flagging might come to the rescue.

Data Quality Dimensions.

A high level of data quality will ensure compliance with legislative direction as well as providing insight into key organisational statistics, allowing informed business planning.
There are six core data quality dimensions we need to be aware of:

Completeness
There are no missing values for a given attribute in the system. For example, in a customer file, there must be a valid value for the "state" field for every customer. In the file for order details, every detail record for an order must be filled.

Uniqueness
The same data must not be stored in more than one place in a system. If, for reasons of efficiency, a data element is intentionally stored in more than one place in a system, then the redundancy must be clearly identified and verified.

Timeliness
The users determine the timeliness of the data. If the users expect customer dimension data not to be older than one day, the changes to customer data in the source systems must be applied to the data warehouse daily.

Validity
Valid if it conforms to the syntax (format, type, range) of its definition

Accuracy
The value stored in the system for a data element is the right value for that occurrence of the data element. If you have a customer name and an address stored in a record, then the address is the correct address for the customer with that name. If you find the quantity ordered as 1000 units in the record for order number 12345678, then that quantity is the accurate quantity for that order.

Consistency
The form and content of a data field is the same across multiple source systems. If the product code for product ABC in one system is 1234, then the code for this product is 1234 in every source system.

Note Space

Explain the nature and challenges of data volumes being processed through integration activities and how a programming approach can improve this.

Big data

Big data consists of both structured and unstructured data.

Structured data is that which can be ordered and processed by data analysis tools. This includes:

➔ Data files organised sequentially or organised serially.

➔ Tables stored within a database management system.

➔ Extensible Markup Language. (XML)

Structured Data is a *specific term* and in a data analytics context should not be thought of the english literal sense of data that has structure. Structured data lends itself to be easily analysed - it is already in the correct format and can be read & processed on mass.

Unstructured data is everything else, e.g.

➔ Word processor, spreadsheet and PowerPoint files

➔ Audio

➔ Video

➔ Sensor and log data

➔ External data (such as social media feeds)

➔ Paper-based documents

Unstructured data is more difficult to analyse - you mostly have to put it in a more structured format, or use something that can apply a methodology (such as indexed tokenization) which will then allow it to be analysed.

Structured data

ID	Name	Age	Degree
1	John	18	B.Sc.
2	David	31	Ph.D
3	Robert	51	Ph.D
4	Rick	26	M.Sc
5	Michael	19	B Sr

Unstructured data

The university has 5600 students.

John's ID is number 1, he is 18 years old and already holds a B.Sc. degree.

David's ID is number 2. he is 31 years old and holds a Ph.D. degree.

Robert's ID is number 3, he is 51 years old and also holds the same degree as David, a Ph.D. degree.

Technical requirements for managing large data sets

The technical requirements for managing large datasets are inherently associated with the challenges faced when dealing with Big data. In particular these are:

Velocity

- ➔ Moving data - big data - is inefficient and slow. No matter how large and fat your pipe is between the data sources and desired destination.The location of data and inertia to movement had to be recognised and respected.
- ➔ This gave rise to the concept of moving distributing your application to where the data is sited, rather than visa versa.

Volume

- ➔ No matter how capable your network is, at Big Data volumes you would quickly run out of bandwidth.
- ➔ Scaling the size of your server gets marginal improvement but at a certain point it becomes too expensive.
- ➔ More importantly, the high-end machines are not cost effective for many applications. For example, a machine with four times the power of a standard PC costs a lot more than putting four such PCs in a cluster.

Veracity

- ➔ If the data becomes corrupt or unavailable (network outage between mass storage device and computer, for example) the expensive and long running query will fail.
- ➔ After many minutes, perhaps hours of running this would be very frustrating
- ➔ We need a more robust solution.
- ➔ Ideally each subset of the data file we have should have multiple copies.
- ➔ If one copy becomes corrupt, there are other copies to fall back on.
- ➔ In other words, we need **data resiliency** and for our system to be **fault tolerant**

Data migration

Data Migration is a process where data is transferred between storage types, formats, data architectures and enterprise systems.

Whereas Data Integration involves collecting data from sources outside of an organization for analysis, migration refers to the movement of data already stored internally to different systems.

Companies will typically migrate data when implementing a new system or merging to a new environment.

Master data management

In business, master data management (MDM) comprises the processes, governance, policies, standards and tools that consistently define and manage the critical data of an organisation to provide a single point of reference.

Integration design

Good data integration design is about creating a single stream of consolidated data. To ensure that this stream of data is accurate, careful planning is required and there are a range of assessments that need to take place before any data integration actually occurs.

As a data analyst, your role is to consider internal and external **rules and regulations**, the **objectives and deliverables** in designing a framework, and the **support models and SLAs** you are committing to within your organisation.

1. Define the project

Setting clear objectives for the project ensures that its success can be measured and monitored. Consider what form the consolidated data has to be in to provide maximum usefulness for your organisation.

An example objective could be daily movement of 100% of sales data from a company's retail outlets to the Customer Relationship Management system at head office, with 98 'up' time.

Define the scope of the project by including its parameters in your objectives and ensure that all relevant databases, datasets and software are listed.

Discover whether there is a need for real-time access to data. If not, how frequently should the data be transferred?

Mitigate risk by listing all potential issues, both during the project's implementation and afterwards, alongside plans to mitigate them. Risks can range from unresponsive system custodians to unreliable data feeds.

2. Understand the systems

Make sure that you review all of the systems involved with the data, from extraction to every system that uses the final, consolidated output. Ensure adequate connectivity between the systems, including any in the cloud, and identify any configurations necessary for the data to be transferred.

Discover whether any manual processes are currently being used, and whether or not these processes can scale. You may also wish to consider whether legacy systems still contribute data and whether these can be replaced by a more modern system.

Data security must be planned for all stages of a data integration. Consider how the transfer of data from each system to another might potentially affect its security.

3. Design the data integration framework

Gain a good understanding of the data to be integrated. For instance, is it structured or unstructured? What are its sources? How good is its quality?

Determine the requirements for the final consolidated data. What datasets are required? How will the data be delivered and how frequently?

List all the data sources needed to meet the project's requirements and how they can be accessed.

Map out the current data flow, if there is one, then determine the data flow required once the project is implemented. Then you will be in a position to define a framework to manage the data flow on an on-going basis.

4. Define how the data will be processed

Where data is supplied by a third party, ensure that the specification of file contents is defined and signed off early. If the data is already available within your organisation, ensure that sample data is available for testing early on in the project.

Profile all the data, whatever its form, to ensure that it meets requirements and that any issues are identified:

- ➔ Dates
- ➔ Monetary values
- ➔ Length
- ➔ Type
- ➔ Format
- ➔ Gaps in the dataset.

You may also need data quality rules to ensure that the data is fit for purpose. Consider:

- ➔ Validation
- ➔ Cleansing
- ➔ Deduplication
- ➔ Consolidation

The reporting requirements identified at Stage 1 will determine whether the data is mapped to new structures or converted into a new format.

Consider how the end user is made aware of whether or not the integration system is working. You may require alerts for system-level issues, such as connectivity, or data-specific issues such as invalid entries. Determine who will receive each type of alert.

5. Implement the project
Choose a project champion to ensure that the integration has a sufficiently high profile within the organisation. He or she can ensure that adequate resources are available and not diverted by other projects.

Identify the stakeholders. Which departments within the organisation use the data or systems and should be involved in the project? Who within those departments will be involved in the design and implementation of the project? Which senior leaders will have input and oversight?

Test the systems thoroughly before implementation, using sample data. This ensures that data quality rules and data mappings have been implemented correctly.

It is important to future-proof the system to avoid dependence on the knowledge of one or two individuals. It's also important that changes or extensions to the data can be made easily. Long and short-term maintenance will be required. You will also need a recovery plan.

Data integration tools (SQL)

Code it yourself vs License an ETL Tool?
There are several advantages and disadvantages however it is best to use an ETL tool if you have a large volume of data to process and it can be quite time consuming if you have a hacked together script doing all the work. However, this costs money and is ultimately down to the business to decide.

Code it yourself

Advantages
➔ Easy to create if the database / data warehouse is small
➔ Easy to install
Disadvantages
➔ Challenging to create especially if the schema changes frequently
➔ Lots of hours to develop scripts
➔ Scalability issues
➔ Requires Programming expertise

License an ETL

Advantages
➔ Company may already have a license
➔ Friendly Graphical User Interface
➔ Support various databases and data formats
➔ Customer support and good documentation
➔ Easy to scale if datasets increase in size
Disadvantages
➔ License costs
➔ Steep learning curve

Data synchronisation

Data Owners are given the right to decide who can have access to enterprise data. They are typically involved in a process that looks like this:

- ➔ A person (staff member, contractor, partner, supplier, etc.) requests access to information
- ➔ A business resource (the Data Owner, the person's manager, etc.) gives the OK
- ➔ A technical resource (usually a DBA) physically grants permission to an application, database, or other data store containing the data. Often, the permission follows a CRUD schema (create, read, update, delete)

Frequency of Updates Task is unlikely to be a one off, so there is a need to assess when new data becomes available, then you can determine how often your scripts need to run. Microsoft has a 'Task Scheduler' through which you can create a batch file.
Performance Scripts should run out of hours to make sure performance is not slowed down
Security Normally when running scripts they will run on business secure servers and not be connected to the internet (unless your data source is online). Some tips on keeping your data secure:

- ➔ Download the data onto a secure server
- ➔ Run ETL on your local file or business databases
- ➔ The Data Owner will issue certain staff with privileges and give you write permissions to the database or you might have to hand over your scripts to the Development team to implement.

Format and data quality Before moving the data into a single point of reference we need to:

- ➔ Remove inconsistencies
- ➔ Standardise various data elements such as naming conventions and codes if our data originates from different datasets
- ➔ Make sure of the meanings of the data names in each file / database / application
- ➔ Deduplication
- ➔ Deriving new calculated values
- ➔ Perform data validation
- ➔ Make sure the data is formatted correctly

Understand testing requirements to ensure that unified data sets are correct, complete and up to date.

Business testing and technical testing

We need testing strategies to make sure that the datasets are:

- ➔ Correct
- ➔ Complete
- ➔ Up to Date

There are 3 types of testing strategies that will be exploring:

1. Technical Acceptance Testing (TAT)

We need to test our scripts to make sure they work and are producing the correct output. We can do this manually but this is time consuming and the tester may forget to do certain checks or there might be human error.
We normally automate these tests to check if the particular input produces an expected output. We may need to create Test Input data sets to accomplish this. The quality of your automated tests are dependent on how well you have written your test scripts.

Unit Tests	Integration Tests	Functional Tests

They consist in testing individual methods and functions of the classes, components or modules used by your script. These tests are automated.
`import unittest`
`def fun(x):`
`  return x + 1`
`class MyTest(unittest.TestCase):`
`  def test(self):`
`  self.assertEqual(fun(3), 4)`
Remember to name your python scripts test_xxx.py
To run:
`python -m unittest`
	Integration tests verify that different modules or services used by your application work well together. For example, it can be testing the interaction with the database.	Functional tests focus on the business requirements of an application. They only verify the output of an action and do not check the intermediate states of the system when performing that action.
		
		There is sometimes a confusion between integration tests and functional tests as they both require multiple components to interact with each other. The difference is that an integration test may simply verify that you can query the database while a functional test would expect to get a specific value from the database as defined by the product requirements.

2. User Acceptance Test (UAT)

UAT enables you to have total confidence in the report you have requested. Acceptance tests are formal tests to verify if a report / system satisfies its business requirements. In order to carry out effective UAT, the person requesting the report needs to develop test scripts to test the following:

- ➔ Does the report meet your original requirements?
- ➔ Does the report produce sensible information e.g. is data duplicated, do known scenarios appear on the report as expected?
- ➔ If it is a formatted report, is the design and layout acceptable?

UAT can be a manual or automatic process to verify the business requirements. Normally a test environment is set up which mimics the production environment, e.g. same version of python, etc.

3. Performance Stress Testing (PST)

A stress test is focused on determining or validating performance characteristics of the product such as scalability and reliability. Performance tests check the behaviours of the system when it is under a significant load.
These tests are based on non functional requirements:

➔ Scalability
➔ Reliability
➔ Stability
➔ Availability

E.g. Observing the response times when executing a high number of data records or seeing how the system behaves with a significant amount of data.

➔ Is the system slow?
➔ Does the system crash?

Note Space

Explain the capabilities (speed, cost, function) of statistical programming languages and software tools, when manipulating, processing and cleaning data and the tools required to solve analysis issues.

Capabilities and functions of statistical programming language

R is a common statistical programming language. It has many built-in functions and libraries, and is extensible, allowing users to define their own functions and procedures using R, C or Fortran. It also has a simple object system.
R provides a wide variety of data types, including vectors (numerical, character, logical), matrices, data frames, and lists. R was designed specifically for statistical computing and data analysis.
Most functionality is provided through built-in and user-created functions and all data objects are kept in memory during an interactive session.
Basic functions are available by default. Other functions are contained in **packages** that can be attached to a current session as needed
R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is **highly extensible**.
At the same time, R also allows data analysts to choose from a wide range of data analysis packages — googleVis, rCharts, gplot2 and ggvis. These data analysis packages make R score over Python. Many data analysts prefer R to Python to visualize data in a more appealing way.
R is an open source tool.

Programming language

Python is a popular programming language used for data analysis. It can do a similar range of tasks as R such as data wrangling, engineering, web scraping etc.
Python is a general-purpose highly flexible programming language can be easily **extended via importing packages/libraries**.
Python provides a range of core data types including: Numbers (Floats & Integers),, Strings, Lists, Dictionaries, Tuples, Files.
Unlike Python, R was not developed as a general-purpose programming language. It was developed for statisticians and data analysts. Hence, programs written in R are generally slower than Python programmes.
Python allows data analysts to choose from several **data visualization libraries — Seaborn, Matplotlib, Bokeh and Altair**. It also offers **data manipulation and cleaning libraries — Pandas** is a popular example. These Python libraries enable users to present huge volumes of data in an easy-to-comprehend visual format.
Python is an open source tool.

Relational databases

SQL (pronounced: “si-kwel”) is short for structured query language. It was developed at IBM by Donald D. Chamberlin and Raymond Boyce in the early 1970s. SQL allows users to access data within a relational database. There are a number of different types of SQL database implementations and platforms, all of which support standard (ANSI) SQL, including:

- ➔ PostgreSQL (open source)
- ➔ MySQL (open source)
- ➔ MS SQL (fee applies)
- ➔ Oracle (fee applies)

Non-relational databases

NoSQL databases (aka "not only SQL") are non tabular, and store data differently than relational tables. NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph. They provide flexible schemas and scale easily with large amounts of data and high user loads.

- ➔ NoSQL is a class of databases built for large amounts of data and fast application speeds
- ➔ Unlike relational databases, NoSQL works without a schema, or set of rules and definitions
- ➔ Most NoSQL databases are built to be resilient and distributed as opposed to transactionally sound

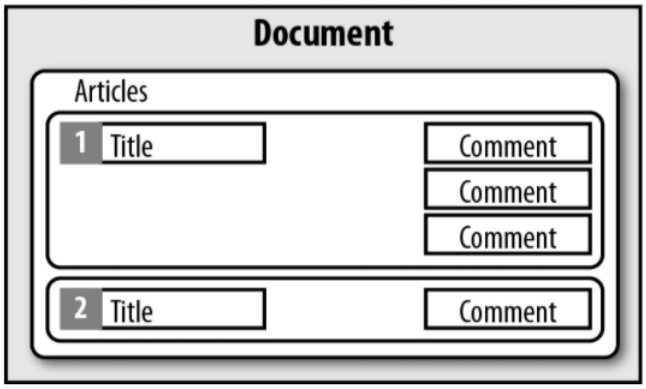
Document

The central concept of a document-oriented database is the notion of a document.

A document structure is just like a key value store, but with *layers*: you can have a value that has keys and other values in it.

While each document-oriented database implementation differs on the details of this definition, in general, they all assume documents encapsulate and encode data (or information) in some standard format or encoding.

Encodings in use include XML, YAML, JSON.



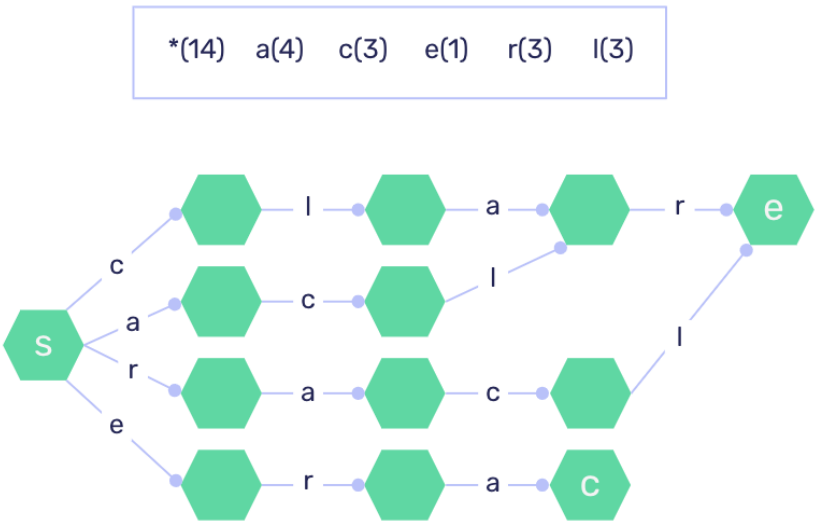
Graph

Graph databases allow you to store entities and relationships between these entities. Entities are also known as nodes, which have properties.

Think of a node as an instance of an object in the application. Relations are known as edges that can have properties.

Edges have directional significance; nodes are organised by relationships which allow you to find interesting patterns between the nodes.

The organisation of the graph lets the data to be stored once and then interpreted in different ways based on relationships.



Key-Value

A key-value store is a simple hash table, primarily used when all access to the database is via primary key.

Think of a table in a traditional RDBMS with two columns, such as ID and NAME, the ID column being the key and NAME column storing the value.

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Column

Column-family stores, such as Cassandra [Cassandra], Amazon SimpleDB [Amazon SimpleDB], allow you to store data with keys mapped to values and the values grouped into multiple column families, each column family being a map of data.

Row-oriented (1)			
name	age	sex	zipcode
thomas	18	male	1416
martin	33	male	1645
bob	25	male	1613

Column-oriented (2)			
name	age	sex	zipcode
thomas	18	male	1416
martin	33	male	1645
bob	25	male	1613

Software tools

Excel is a local tool that is suitable for dealing with generally smaller datasets. It has a fixed upper limit of 1,048,576 rows and 16,384 columns and the software's performance is limited by your computer's available memory and system resources. Excel offers similar functions as SQL, including methods such as COUNT, SUM, AVG, IF THEN ELSE, etc.

It can perform basic data analysis including statistical modeling, visualisations and reports, however is not well suited for advanced applications such as machine learning.

Excel is not an open source software.

Note Space

Explain how statistical programming languages are used in preparing data for analysis and within analysis projects.

Preparation techniques

Statistical programming languages are often used in preparing data for analysis by providing a set of typical functions that allow the user to familiarize themselves with the data and separate out data that are relevant and valuable to their projects. Some of these functions include:

- Searching and sorting: refers to familiarisation of yourself with data and their high level characteristics, as well as sorting them according to your project's relevancy.
- Grouping: is the formation of groups of data that follow specific rules or share specific characteristics.
- Filtering: is the creation of a subset using simple operator comparisons performed on columns to extract relevant or drop irrelevant information. For example, you can filter out the rows in a column that obey a condition.
- Modelling: is the final transformation in preparing data for analysis and helps transform the data into a target state. Statistical programming languages enable modeling with functions such as normalisation and scaling of variables, discretisation, smoothing and attribute construction, among others.

Data cleaning to remove a range of data issues

A data analyst often has to deal with a range of data errors such as:
Data Missing – e.g. only a portion of the data set is available.
Data Redundancy - e.g repeated data within the same dataset
Data Inconsistencies – e.g. wrong data types, wrong encoding, wrong/symbols, commas in the wrong place.
Invalid values is another cause of issues in preparing data for analysis. For example having a numerical value instead of a string value under the *'name'* column of a dataset.
Out of range data value error occurs when the data value is outside a predefined interval of data. For example if you have a column that takes values from 1 to 5, a data value of 0 or 6 would be an example of out of range data.
Outliers: are the data values that deviate significantly from other data values present in the dataset. Strictly speaking, any value that lies outside the range of three standard deviations from the mean of your data set, is considered an outlier.

Processing and analysing

The following descriptive statistics definitions represent measures of central tendency, i.e. the the centre point or typical value of a dataset.

- **Mean:** Average or expected value. It is determined by summing all data points in a population and then dividing the total by the number of points.
- **Median:** The median refers to the midpoint in a series of numbers.
- **Mode:** The mode of a set of values is the value that occurs most often. A set of values may have more than one mode, or no mode at all.
- **Range**: The range of a distribution with a discrete random variable is the difference between the maximum value and the minimum value.

Probability is defined as the likelihood of one outcome occurring over the set of all possible outcomes:

$$probability = \frac{one\ outcome}{all\ outcomes}$$

(Sampling) bias

Sampling bias occurs when a sample is collected in such a way that some members of the intended population are more or less likely to be included than others.

This can happen when a sample is taken non-randomly — either implicitly or explicitly.
When we have non-random sampling that results in sampling bias, it can affect the inferences or results of our analyses. We must be sure not to attribute our results to the process we observe when they could actually be because of non-random sampling.

Conceptually, this is straightforward: When we have sampling bias, we aren't measuring what we think we are measuring.

Examples of Sampling Bias

- ➔ **Pre-screening:** Purposely restricting the sample to a specific group or region.
  - This typically happens when people try to study priority areas to save costs and assume priority areas are the same as random areas.
- ➔ **Self-selection:** When someone has the ability to non-randomly decide what is included in a sample.
  - This typically happens in surveys and polls but can also be an issue with other kinds of reporting.
- ➔ **Survivorship bias:** When we select only surviving subjects in a sample over time.
  - This might happen when we only look at existing customers and assume they have the same characteristics as new customers.

Problems That Arise From Sampling Bias

- ➔ We could overestimate or underestimate means and sample statistics for simple characteristics.
- ➔ It's possible to have artificial correlation where there should be none.

Recovering From Sampling Bias

- ➔ Working out causal DAGs can help you identify when to watch out for sampling bias.
- ➔ Generally, it's best to prevent sampling bias whenever possible.
- ➔ We can't really do anything if we ENTIRELY exclude an important group of data.
- ➔ However, if portions of our data are overrepresented or underrepresented, there are ways to correct that effect.
  - Typically, we explicitly model the selection process, which means we need data on factors that determine whether or not someone participates.

Statistical significance

Statistical significance indicates our degree of confidence to infer a parameter about the overall population from a statistic calculated in our sample. Statistical significance is often linked to a *p-value*.
The *p-value* is the probability that, given the null hypothesis (H0) is true, we could have ended up with a statistic at least as extreme as the one measured from our random sample of data from the true population.
It's common to see Frequentist tests reported as "*statistically significant with p < 0.05" or "with p < 0.01*." So, what does it mean for a test to be statistically significant? On the surface, these statements are simply saying that the calculated p-value is less than a specific value. The values of 0.05 and 0.01 are common in academic research but also arbitrary.
What does having "p < 0.05" specifically mean?

- ➔ **p < 0.05:** In hypothetical repetitions of this experiment with the same sample size, fewer than 5% of the experiments would have measured a difference between arms at least this extreme by chance.

The same goes for "p < 0.01." Here, however, it's framed in the context of null and alternative hypotheses:

- ➔ **p < 0.01:** There is less than a 1% chance of accepting the alternative hypothesis when the null hypothesis is in fact true.

Linear regression (simple and multiple):

In simple linear regression, we predict scores on one variable from the scores on a second variable. The variable we are predicting is called the criterion variable and is referred to as Y.
The variable we are basing our predictions on is called the predictor variable and is referred to as X.
When there is only one predictor variable, the prediction method is called simple regression. In simple linear regression, the topic of this section, the predictions of Y when plotted as a function of X form a straight line.
Similarly, if there is more than one predictor variable the prediction method is called multiple regression.

Advantages of linear regression	Disadvantages of linear regression
➔ Simple to explain. ➔ Highly interpretable. ➔ Model training and prediction are fast. ➔ No tuning is required (excluding regularization). ➔ Features don't need scaling. ➔ Can perform well with a small number of observations. ➔ Well understood.	➔ Presumes a linear relationship between the features and the response. ➔ Performance is (generally) not competitive with the best supervised learning methods due to high bias. ➔ Can't automatically learn feature interactions.

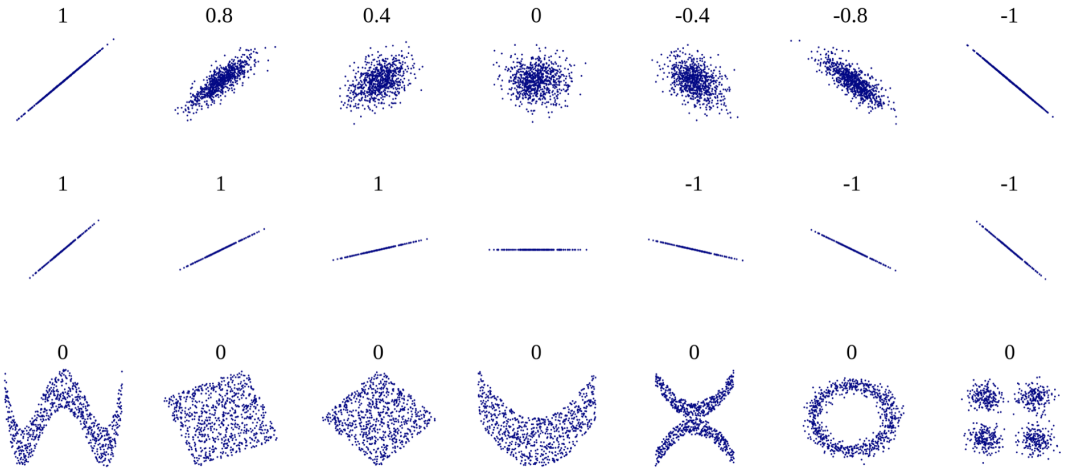
Scatter plots and correlation

Correlation measures how variables are related to each other.

Typically, we talk about the Pearson correlation coefficient — a measure of linear association. We refer to perfect correlation as *collinearity*.

The following are a few correlation coefficients. Note that if both variables trend upward, the coefficient is positive. If one trends opposite the other, it is negative.

It is important that you always look at your data visually (scatterplots) — the coefficient by itself can be misleading:



And / Or probability

'AND' Rule:

When you want to estimate the probability of two or more events (outcomes) happening together, you assume the 'AND' rule. This means that the 'AND' rule, returns the probability that all events (outcomes) are taking place together. The probability p of events A **and** B happening, is given by

$$p(A \text{ and } B) = p(A) \times p(B)$$

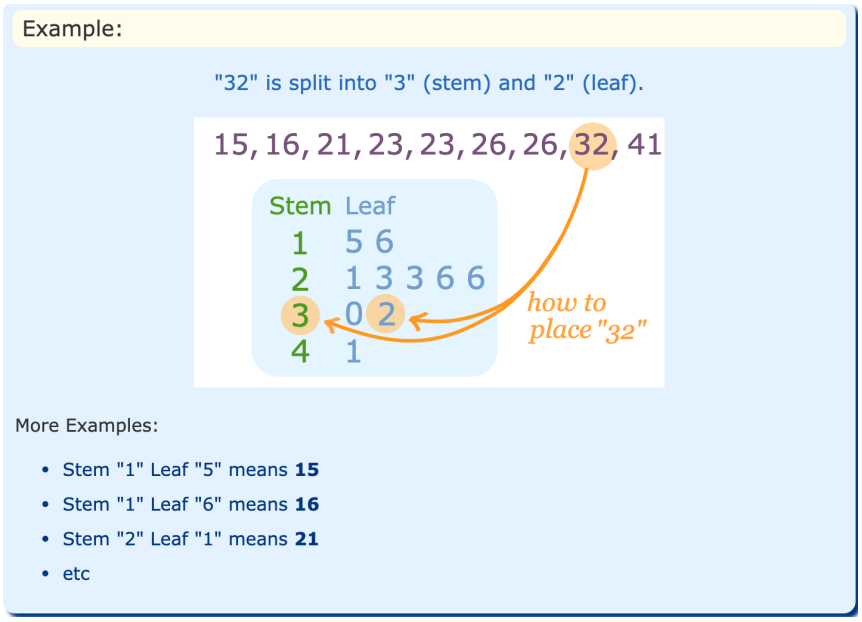
where $p(A)$and $p(B)$ are the probabilities of events A and B, respectively.

'OR' Rule:
When you want to estimate the probability of one or another event happening, you assume the 'OR' rule. This means that the 'OR' rule, returns the probability that either one of the events are occuring. The probability p of event A **or** event B happening, is given by

$$p(A \text{ or } B) = p(A) + p(B)$$

Stem and leaf plots (frequency and distribution)

A **Stem and Leaf Plot** is a special table where each data value is split into a "stem" (the first digit or digits) and a "leaf" (usually the last digit). Like in this example:



The "stem" values are listed down, and the "leaf" values go right (or left) from the stem values.

The "stem" is used to group the scores and each "leaf" shows the individual scores within each group.

Stem and leaf plots are used to display quantitative data (i.e. numbers) for small data sets of no more than 40 or 50 points. They display the exact values of each data point. These plots are useful for visualizing frequency distributions in data sets.

Factorials

Factorial is the mathematical operation of performing a multiplication between all descending sequential numbers, starting with the number we wish to find the factorial of and 'descending' to 1.

A factorial is notated with an exclamation mark (!) after a number. For example if you want the factorial of number 10 (noted as $10!$), then:

$$10! = 10 \times 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 3628800$$

similarly, if you want to calculate the factorial of number 5 (noted as $5!$) then:

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

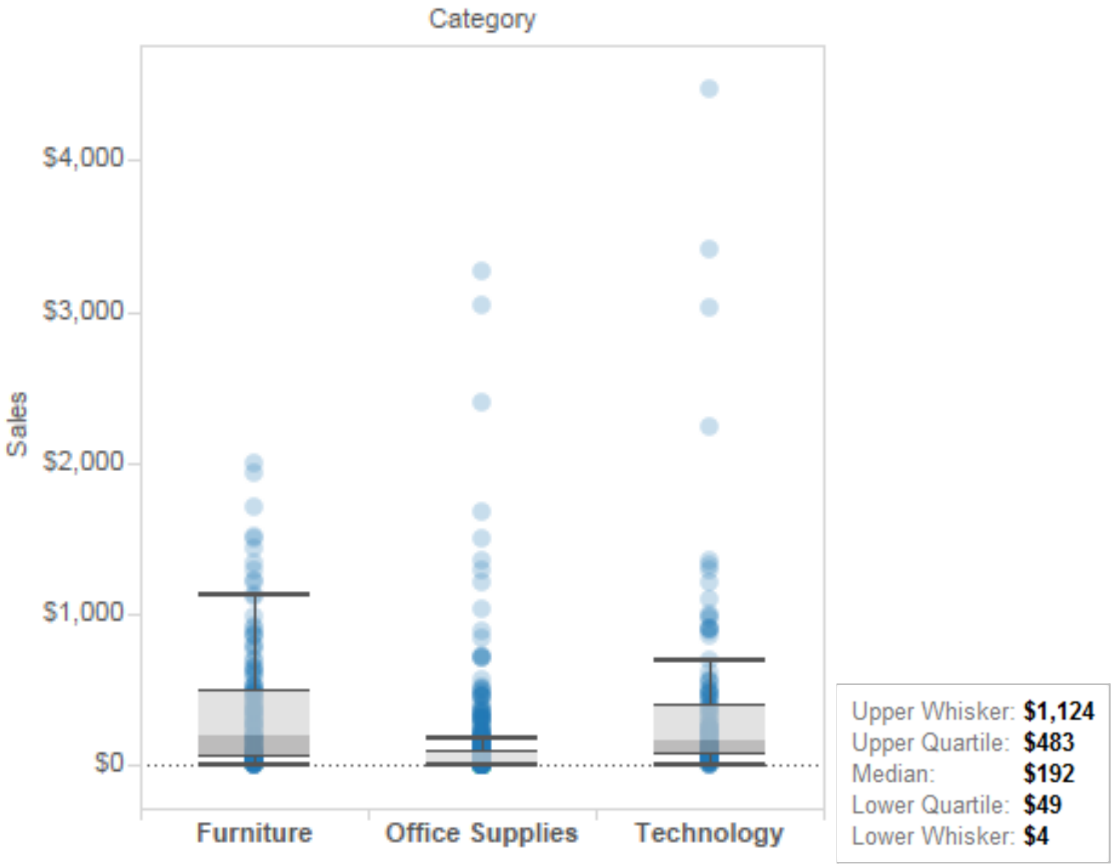
Factorials find extensive use in specific areas of probability theory such permutations.

Box and whisker plots

Box-and-whisker plots ("box plots") create a box around the second and third quartile of a range of disaggregated data points. Box plots have the following properties:

- ➔ effective choice for visualizing distributions.
- ➔ different from histograms because the "bins" aren't defined as set sizes, but instead by the 25th, mean, and 75th percentiles.
- ➔ great for showing relative differences between the distribution of data points for different groups.
- ➔ The bottom and top "whiskers" extend to the first and fourth quartiles.
- ➔ Outliers are disregarded in terms of finding the middle 50 situated around the median.

The following figure is an example of a box-and-whisker plot.



Methods for presenting results

In addition to considering data visualization attributes, you should carefully choose the type of chart or graph you'll use. Let's look at a few commonly used charts and graphs.

Tables:

A **pivot table** brings together, simplifies and summarizes information stored in other tables and spreadsheets, stripping this down to the most pertinent insights. They are also used to create unweighted cross tabulations fast. Pivot tables are one of the most simple and useful ways to visualize data. That's because they allow you to quickly summarize and analyze large amounts of data, and to use additional features such as color formatting and data bars to enhance the visual aspects.

Pivot tables are more about simplifying tables than changing it into a graphical representation. That means they are helpful for displaying data with several subcategories in easily digestible ways.

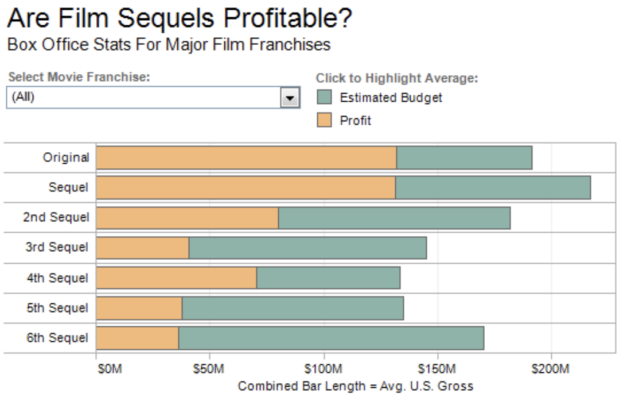
DIAGNOSIS	# PATIENTS	AVG COST	AVG STAY (DAYS)
Bypass	116	$781,957	8.07
Cardiac Arrest	120	$796,833	8.92
Chemotheraphy	119	$804,966	9.17
Chronic Headache	122	$780,387	10.03
Diabbetes	154	$788,317	7.88
Ear infection	122	$764,097	6.81
EKG	131	$790,577	8.63
Epilepsy	110	$829,782	8.70
Hypoglycemia	125	$803,405	7.93
Radiotheraphy	168	$768,751	8.45
Grand Total	199	$788,621	8.42

Age Range		Total Quantity
0-18	New	1,209
	Refurbished	292
	Unspecified	360
	Used	1,913
19-24	New	2,544
	Refurbished	592
	Unspecified	569
	Used	3,790
25-34	New	5,950
	Refurbished	1,401
	Unspecified	1,407
	Used	8,831

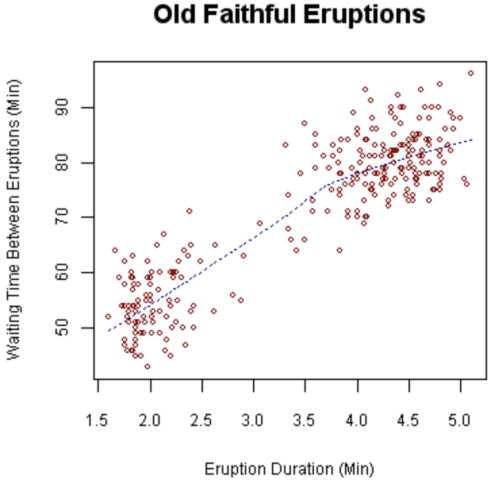
Source: Sisense.com

Charts:

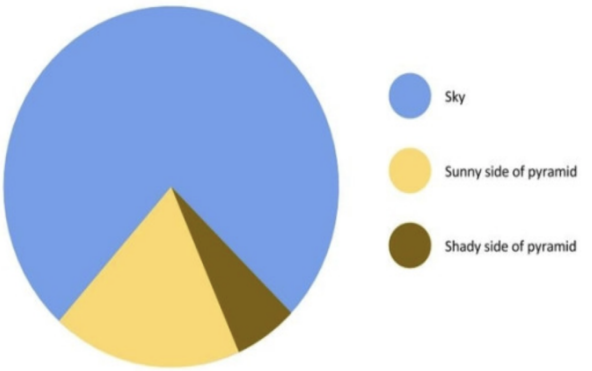
Bar chart: Bar charts are one of the most common ways of visualizing data. Why? Because they make it easy to compare information, revealing highs and lows quickly. Bar charts are most effective when you have numerical data that splits neatly into different categories.



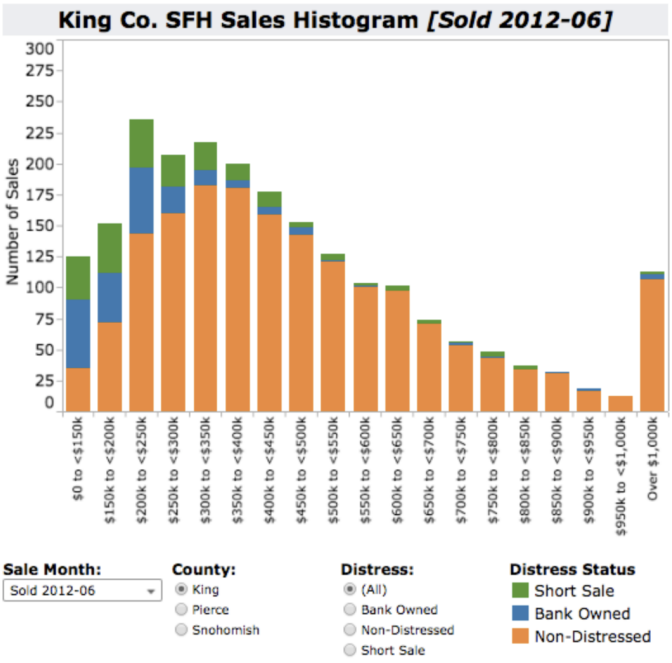
Scatterplot: Scatter plots are a great way to give you a sense of trends, concentrations, and outliers. This will provide a clear idea of what you may want to investigate further.



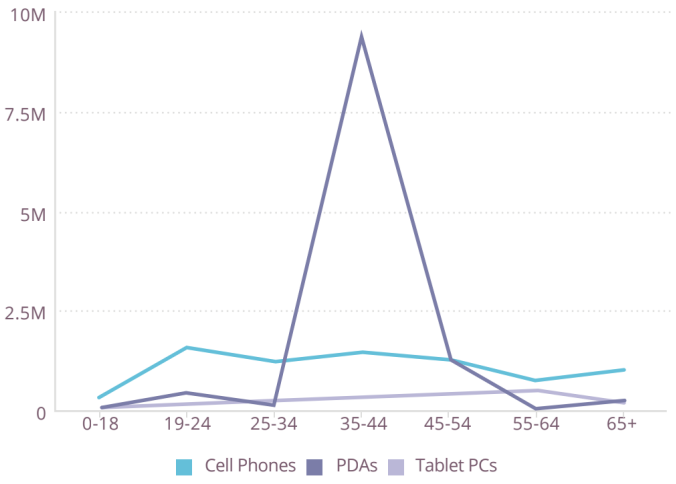
Pie chart: Pie charts are the most commonly misused chart type. They should be only used to show relative proportions or percentages of information. If you want to compare data, leave it to bars or stacked bars. If your viewer has to work to translate pie wedges into relevant data or compare pie charts to one another, the key points you're trying to convey might go unnoticed.



Histograms: Histograms are useful when you want to see how your data are distributed across groups. A common pattern is the bell-shaped **curve** known as the "**normal distribution**." In a **normal** or "typical" **distribution**, points are as likely to occur on one side of the average as on the other. Note that other **distributions** look similar to the **normal distribution**. **Skewness** is the measure of the asymmetry of a **histogram** (frequency distribution). A **histogram** with normal distribution is symmetrical. In other words, the same amount of data falls on both sides of the mean. A normal distribution will have a **skewness** of 0. **Kurtosis** is a measure of the combined weight of the tails in relation to the rest of the distribution. As the tails of a distribution become heavier, the **kurtosis** value will increase. As the tails become lighter the **kurtosis** value will decrease. A **histogram** with a normal distribution has a **kurtosis** of 0.



Line chart: Line graphs plot data points on a graph and then join them up with a single line that zigzags from each point to the next. You can also compare changes over the same period of time for more than one group or category very easily, by adding a “break by” category.
There are many different business cases that work well with line charts. Pretty much anything that compares data, or shows changes, over time is well suited to this type of visualization. Again, it’s all about visualizing a trend. You can also compare changes over the same period of time for more than one group or category very easily, by adding a “break by” category.
Source: Sisense.com



This diagram shows some general guidelines on what chart to use when: