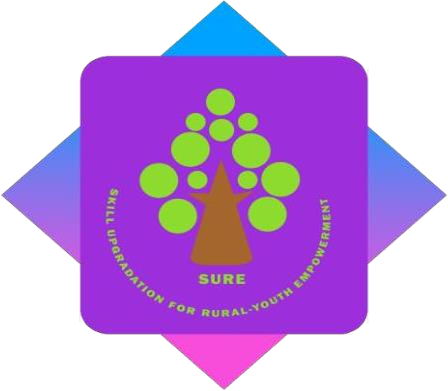
**SQL & Microsoft Power Bi Tools**

**A CERTIFICATE COURSE CONDUCTED**

**BY**



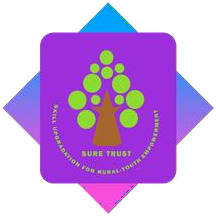
## THE SURE TRUST

**Skill Upgradation for Rural-youth Empowerment – TRUST (**[**www.suretrustforruralyouth.com**](http://www.suretrustforruralyouth.com/)**)**

**COURSE TRAINING ATTENDED BY**

**A Noorjahan Begum**

**December 2022- March 2023**

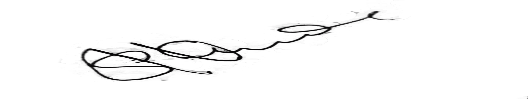


Declaration

This is to certify that Ms A. Noorjahan Begum has successfully completed the four months training in “Power BI and SQL” course conducted by SURE Trust during the period December 2022 - March 2023.

By

Uma Mahesh R

Senior Manager at Novac Tech, Chennai, India

Trainer in SURE Trust



Prof. Radhakumari Mrs.

Vandana Nagesh

Executive Director & Founder Director & Founder

SURE Trust SURE Trust

**TABLE OF CONTENTS**

1. The SURE TRUST
2. Course Content
3. Conduct of the Course:
   * Student byelaws
   * Written Tests
   * Assignments
4. Student Feedback:
5. Uniqueness of the Course according to student:
6. Concluding Remarks:
7. **Introduction to the SURE TRUST**

Introduction to The SURE TRUST:

The SURE TRUST is born to enhance the employability of educated unemployed rural youth. It is observed that there is a wide gap between the skills acquired by students from the academic institutions and the skills required by the industry to employ them. Employability enhancement is done through giving one on one training in emerging technologies, completely through online mode. The mission of the SURE TRUST is to bridge the gap between the skills acquired and the skills required by training them in the most emerging technologies such as Artificial Intelligence (AI), Python Program, Machine Learning (ML), Deep Learning (DL), Data Science & Data Analytics, Blockchain Technology, Robotic Process Automation (RPA), Project Management, Excel for Business Application, Statistical tools & Applications, Spoken English and Business Communication etc., that will enhance their employability. After completion of four months training in the course, the trainees will get live projects from industries as internship activity to get experience in applying to real time situation what they have learnt during the course. These projects will give them hands on experience which is much sought after by the prospective industry employing them. Currently students from all over India are enrolling for various courses offered by the SURE TRUST. The SURE TRUST offers every course free of cost with no financial burden of any kind to students. This initiative is purely a service-oriented one aiming to guide the rural youth who are educated but unemployed due to lack of upgradation in their skill sets. The birth of SURE TRUST is a God given boon to rural youth who could reach great heights either in employment or in entrepreneurship once they receive the training offered followed by the company internship. Many companies are coming forward to join their hands with us by offering internship projects to hand hold and lead the rural youth in their career settlement.

Vision of the SURE TRUST:

The vision of the SURE TRUST is to enhance the employability of educated unemployed youth, particularly living in rural areas, through skill upgradation, with no cost to the students.

Mission of the SURE TRUST:

The mission is to bridge the gap between the skills acquired in the academic institutions and the skills required in industries as a pre-condition for employment.

Functioning of the SURE TRUST:

There are three dedicated, committed, and hard-working women on the board of management of the SURE TRUST who will look into the various administrative and other matters relating to the enrolment of students, organizing trainers, entering into agreements with companies for getting live projects to students as internship programs, and so on. All the three women on the board are all the alumni from Sri Sathya Sai Institute of Higher Learning, Anantapur Campus, deemed to be a University. The women board is supported by five eminent advisories who are from different walks of life and have made outstanding mark in career in their respective fields. For more details about SURE TRUST please visit the website [www.suretrustforruralyouth.com](http://www.suretrustforruralyouth.com/)

1. Course Content:

The SURE TRUST conducts a four months training for every course on a uniform basis. A session spanning across one to one & half hour is taken by the trainers for every major course. Sessions are conducted to complete the predesigned course structure within the fixed time period. Course content is designed to suit the current requirement of the Industry and validated by the industry experts. The course content of all these courses is so dynamic that any changed condition noticed in the industry will automatically get reflected in the content of the respective course. As the course content is dynamic, the Following is the course content of the current course in SQL Microsoft Power Bi Tools

**SQL & Microsoft Power Bi Tools**

**About Power BI**

**Power BI:**

Power BI is a powerful business analytics service provided by Microsoft that enables users to visualize and analyse data from a variety of sources in a user-friendly and interactive way. It allows users to create interactive reports, dashboards, and data visualizations, which can be shared with others and accessed on various devices.

Power BI integrates with many different data sources, including spreadsheets, databases, cloud services, and big data sources. It provides a wide range of data transformation and manipulation options, such as data cleaning, merging, and shaping, which can help users prepare their data for analysis.

The Power BI interface is intuitive and easy to use, even for non-technical users. Users can create interactive dashboards with drag-and-drop functionality, custom visuals, and various filters, slicers, and drill-down options. They can also share their reports and dashboards with others, either by publishing them to the cloud or embedding them in websites or other applications.

Power BI offers a variety of advanced features for enterprise-level data analytics, including data modelling, advanced data visualizations, and collaboration tools. It also provides extensive security and compliance features to ensure that data is kept secure and private.

Overall, Power BI is a versatile and powerful tool for businesses of all sizes that need to analyse and make sense of their data. It can help users gain valuable insights, make better decisions, and improve their overall business performance.Top of Form

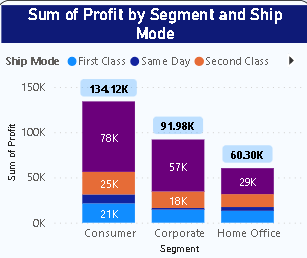
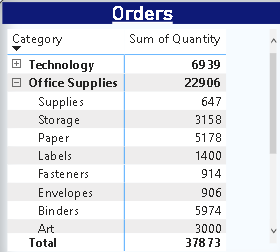
**More About Power BI:**

Power BI is a powerful business intelligence and data visualization tool that enables users to connect to a wide variety of data sources and create visually appealing reports and dashboards. Some of the key features and capabilities of Power BI include:

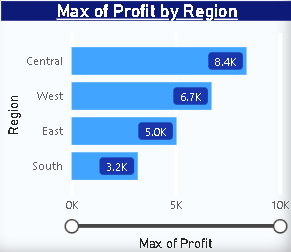
1. Data Connectivity: Power BI can connect to a wide variety of data sources, including popular cloud-based services such as Azure, Google Analytics, Salesforce, and others. It also connects to on-premises data sources such as SQL Server, Oracle, Excel, and many more.
2. Data Transformation and Shaping: Power BI provides users with a range of data transformation and manipulation tools that help them clean, shape, and transform their data into a more usable format. These tools include options for data cleaning, merging, and appending.
3. Interactive Data Visualizations: Power BI allows users to create interactive reports and dashboards that can be customized to fit their specific needs. Users can choose from a variety of built-in visuals or create their own custom visuals.
4. Collaboration and Sharing: Power BI makes it easy for users to collaborate on reports and dashboards by allowing them to share their work with others. Users can share their reports and dashboards with colleagues or external stakeholders, either by publishing them to the cloud or embedding them in a website or other application.
5. Advanced Analytics: Power BI provides users with advanced analytics capabilities such as forecasting, clustering, and anomaly detection. These features enable users to gain valuable insights from their data and make data-driven decisions.
6. Security and Compliance: Power BI provides extensive security and compliance features to ensure that data is kept secure and private. It also supports a variety of compliance standards such as HIPAA, GDPR, and ISO 27001.

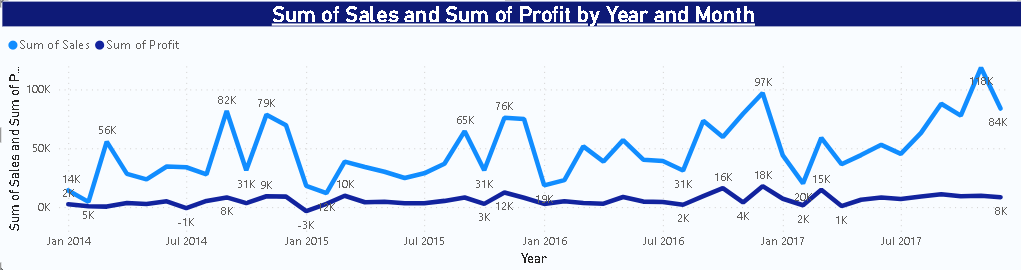
Overall, Power BI is a versatile and powerful tool for businesses of all sizes that need to analyze and make sense of their data. It can help users gain valuable insights, make better decisions, and improve their overall business performance.

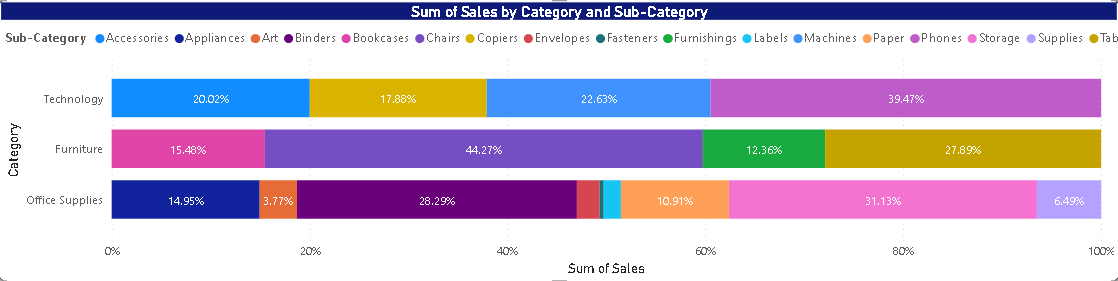
Top of Form

**Assignment-1**

**(Visualizing data)**

****





**Assignment-2**

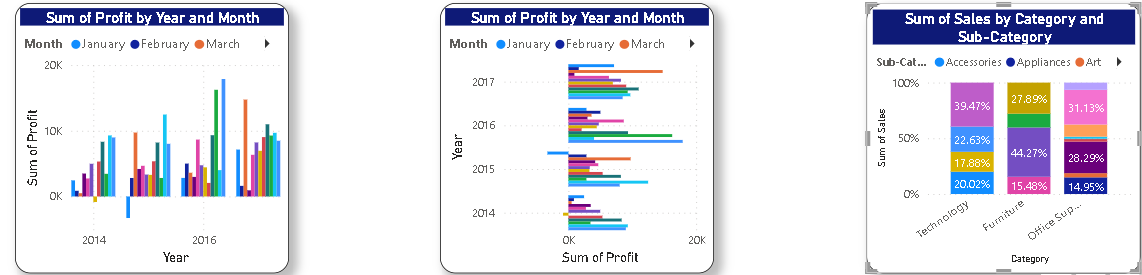
**Types of Graphs available in Power BI and their purposes:**

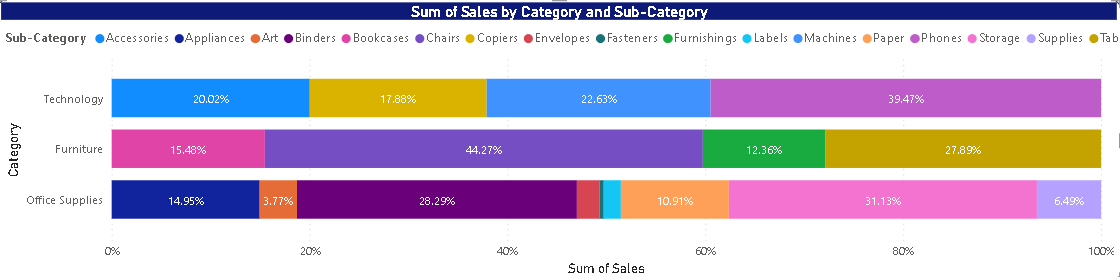
1. Column Chart: A column chart is a vertical bar chart used to show comparisons among categories. It is used to compare data across categories over time.
2. Line Chart: A line chart is used to show trends over time. It is a powerful tool for visualizing continuous data over time.
3. Pie Chart: A pie chart is a circular chart used to show the composition of something. It is used to show how much each category represents as a part of a whole.
4. Bar Chart: A bar chart is a horizontal chart used to show comparisons among categories. It is used to compare data across categories.
5. Scatter Chart: A scatter chart is used to plot two-dimensional data points on a graph. It is used to show the relationship between two variables.
6. Bubble Chart: A bubble chart is similar to a scatter chart but uses bubbles to represent data points. The size of the bubble represents the value of a third variable.
7. Waterfall Chart: A waterfall chart is used to show how an initial value is affected by a series of intermediate positive or negative values.
8. Treemap: A treemap is a hierarchical chart used to show the proportions of different categories in a dataset. It is used to visualize the relative sizes of different categories in a dataset.
9. Gauge Chart: A gauge chart is used to show the progress towards a specific goal or target. It is used to visualize the percentage of completion of a target.
10. Funnel Chart: A funnel chart is used to show how a process or system is progressing through stages. It is used to visualize the conversion rates of a process.
11. Area Chart: An area chart is a line chart with the area below the line filled in with color. It is used to show the trend of a data series over time and is often used to compare two or more data sets.
12. Donut Chart: A donut chart is similar to a pie chart but with a hole in the center. It is used to show the composition of something, and the size of each slice represents the proportion of that category.
13. Radar Chart: A radar chart is a two-dimensional chart that is used to display multivariate data. It is used to compare several quantitative variables for a single data point.
14. KPI Chart: A KPI (Key Performance Indicator) chart is used to measure the progress towards a specific goal or target. It is used to visualize the current status of a metric, such as revenue, profit, or customer satisfaction.
15. Histogram: A histogram is a graphical representation of a frequency distribution of a dataset. It is used to show the distribution of numerical data.
16. Box and Whisker Plot: A box and whisker plot is used to show the distribution of a dataset. It is used to visualize the spread and skewness of the data.
17. Map: A map is used to visualize geospatial data. It can display data such as location, density, and regions.
18. Sunburst Chart: A sunburst chart is a circular chart that is used to display hierarchical data. It is used to visualize the proportion of different categories within a hierarchy.
19. Stacked Column Chart: A stacked column chart is a variation of a column chart that is used to show how much each category represents as a part of a whole. It is used to compare data across categories and over time.
20. Stacked Bar Chart: A stacked bar chart is a variation of a bar chart that is used to show how much each category represents as a part of a whole. It is used to compare data across categories.
21. Bullet Chart: A bullet chart is used to show progress towards a specific goal or target. It is similar to a gauge chart but allows for more detailed visualization of the data.
22. Dual-axis Chart: A dual-axis chart is used to display two different types of data on the same chart. It is useful when there are two different scales or units of measure.
23. Combo Chart: A combo chart is used to display two or more different types of data on the same chart. It is useful when there are two different scales or units of measure.
24. Heat Map: A heat map is used to show the distribution of numerical data using colors. It is useful for visualizing data that have a large number of data points.
25. Area Chart with Y-axis: An area chart with Y-axis is similar to an area chart but includes a secondary Y-axis. It is used to visualize two data series that have different scales.
26. Gauge: A gauge is used to show progress towards a specific goal or target. It is useful when the data can be measured in percentages or values.
27. Ribbon Chart: A ribbon chart is used to display changes over time. It is useful for visualizing trends and patterns in data.
28. Tree Map: A tree map is used to visualize hierarchical data. It is similar to a treemap but shows the hierarchy of the data in a more structured way.
29. Waterfall Chart with Subtotal: A waterfall chart with subtotal is similar to a waterfall chart but includes subtotals for each category. It is useful for visualizing how each category contributes to the overall total.
30. Multi-row Card: A multi-row card is used to display multiple pieces of information about a single data point. It is useful when there are several pieces of data that need to be displayed together.
31. Funnel Chart: A funnel chart is used to visualize the different stages of a process. It is used to show how many leads are converted to customers at each stage of a sales funnel.
32. Scatter Chart: A scatter chart is used to visualize the relationship between two numerical variables. It is used to see if there is a correlation between the two variables.
33. Treemap: A treemap is used to visualize hierarchical data. It is used to display the proportions of different categories within a hierarchy.
34. Pie Chart: A pie chart is used to show the composition of something. The size of each slice represents the proportion of that category.
35. Column Chart: A column chart is used to compare data across categories. It is used to display data in vertical columns.
36. Bar Chart: A bar chart is used to compare data across categories. It is used to display data in horizontal bars.
37. Line Chart: A line chart is used to show the trend of a data series over time. It is used to visualize changes in data over time.
38. 100% Stacked Column Chart: A 100% stacked column chart is used to compare data across categories, and show how much each category represents as a part of a whole. It is used to visualize the proportion of each category as a percentage of the whole.
39. 100% Stacked Bar Chart: A 100% stacked bar chart is used to compare data across categories, and show how much each category represents as a part of a whole. It is used to visualize the proportion of each category as a percentage of the whole.
40. Waterfall Chart: A waterfall chart is used to visualize the cumulative effect of positive and negative values. It is used to show how an initial value is affected by a series of intermediate positive or negative values.

Top of Form

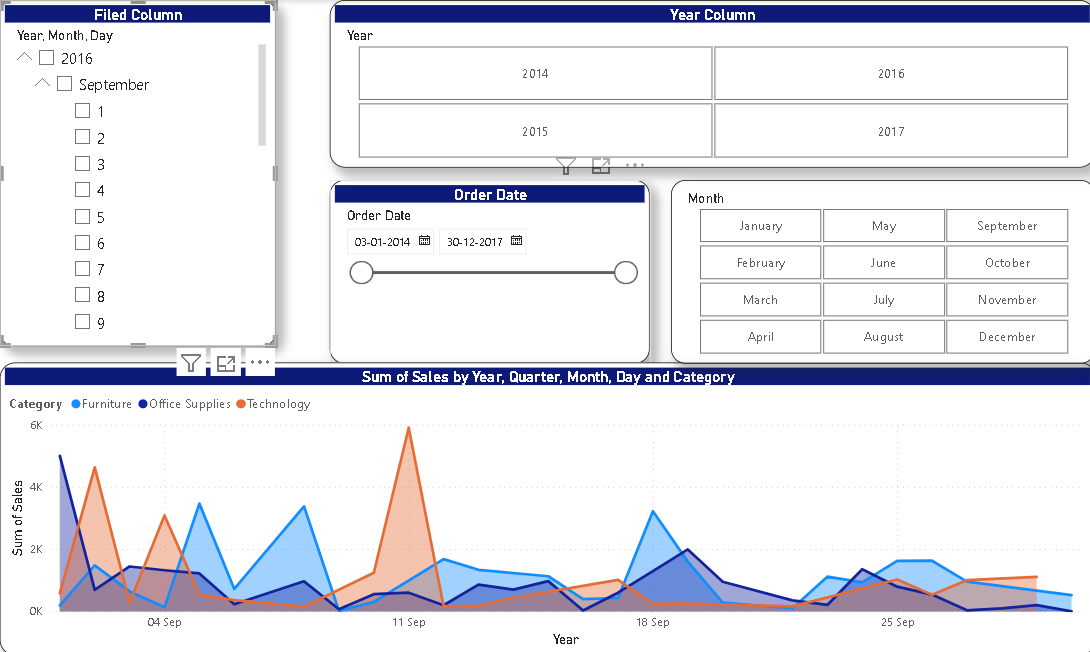
1. Donut Chart: A donut chart is similar to a pie chart but has a hole in the center. It is used to show the composition of something, just like a pie chart.
2. Scatter and Bubble Chart: A scatter and bubble chart is used to visualize the relationship between two numerical variables. It is similar to a scatter chart but includes a third variable that is represented by the size of the bubbles.
3. KPI Chart: A KPI chart is used to display a single metric that is used to evaluate the performance of a company or organization. It is used to highlight the performance of a company or organization against a target or benchmark.
4. Gauge Chart with Variance: A gauge chart with variance is similar to a gauge chart but includes a variance that shows the difference between the actual value and the target value.
5. Map: A map is used to display data geographically. It is used to visualize data that is related to specific locations.
6. Card: A card is used to display a single metric. It is used to highlight a specific piece of information.
7. Table: A table is used to display data in a tabular format. It is used to display multiple metrics across different categories.
8. Ribbon Chart with Forecast: A ribbon chart with forecast is similar to a ribbon chart but includes a forecast that shows the expected future values of the data series.
9. Stacked Column Chart: A stacked column chart is used to compare data across categories. It is used to display data in vertical columns, with each column representing a different category and the height of each column representing the total value of the data series.
10. Stacked Bar Chart: A stacked bar chart is used to compare data across categories. It is used to display data in horizontal bars, with each bar representing a different category and the width of each bar representing the total value of the data series.
11. Dual Axis Chart: A dual axis chart is used to compare two data series that have different scales. It is used to display the two data series on different axes.
12. Histogram: A histogram is used to visualize the distribution of a continuous variable. It is used to group the data into bins and display the frequency of each bin.
13. Box and Whisker Plot: A box and whisker plot is used to visualize the distribution of a continuous variable. It is used to display the median, quartiles, and range of the data series.
14. Area Chart: An area chart is used to show the trend of a data series over time. It is used to visualize changes in data over time, similar to a line chart, but the area below the line is filled with color.
15. Heatmap: A heatmap is used to visualize the relationship between two categorical variables. It is used to display the frequency of the combinations of the two categorical variables.
16. Bullet Chart: A bullet chart is used to display a single metric, such as a KPI or a performance metric. It is used to highlight the performance of a metric against a target or benchmark.
17. Tree Map: A tree map is used to visualize hierarchical data. It is used to display the proportions of different categories within a hierarchy, similar to a treemap, but with rectangles instead of squares.
18. Combo Chart: A combo chart is used to display multiple data series on the same chart. It is used to compare multiple data series that have different scales.
19. Polar Chart: A polar chart is used to display data that is distributed around a circle. It is used to visualize the relationship between different categories around a circle.
20. Sunburst Chart: A sunburst chart is used to visualize hierarchical data. It is used to display the proportions of different categories within a hierarchy, similar to a treemap or tree map, but with a circular layout.

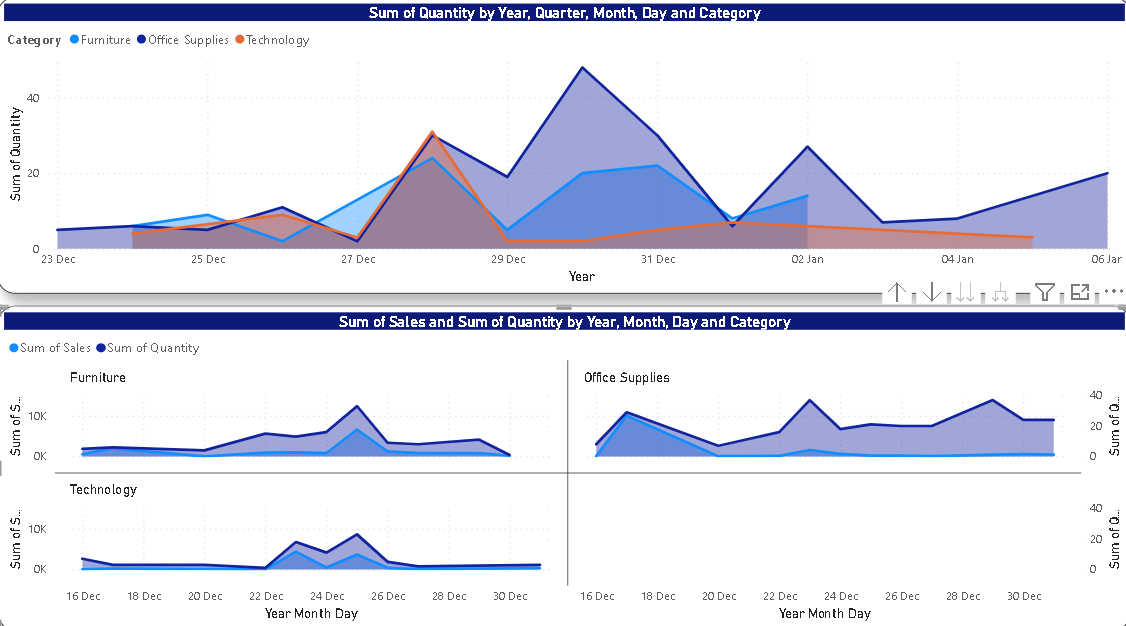
Top of Form

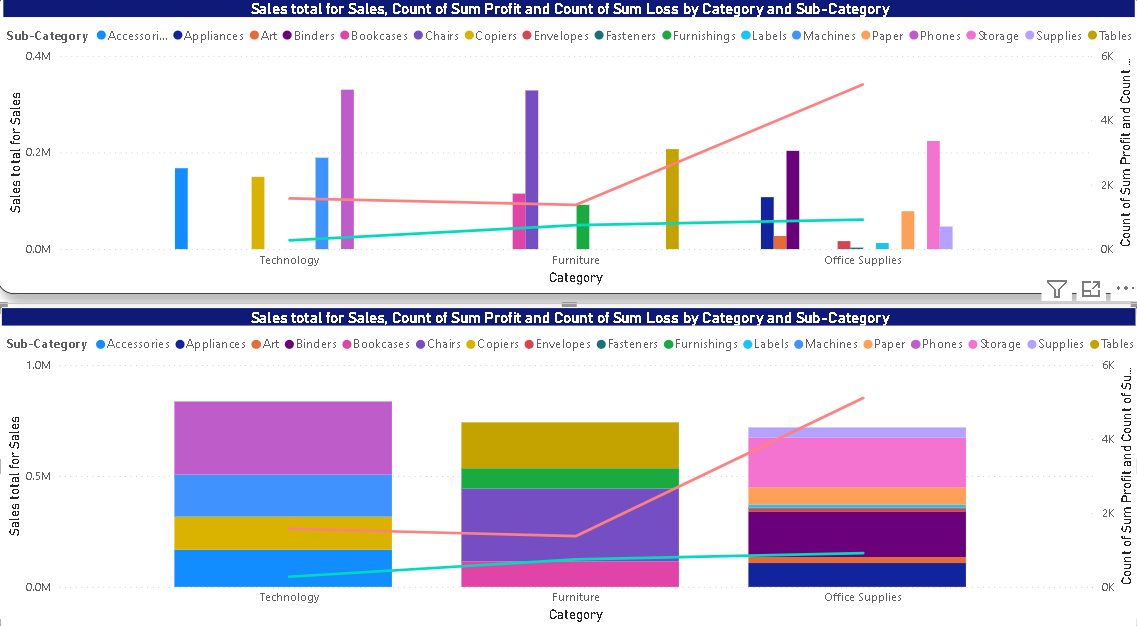
****

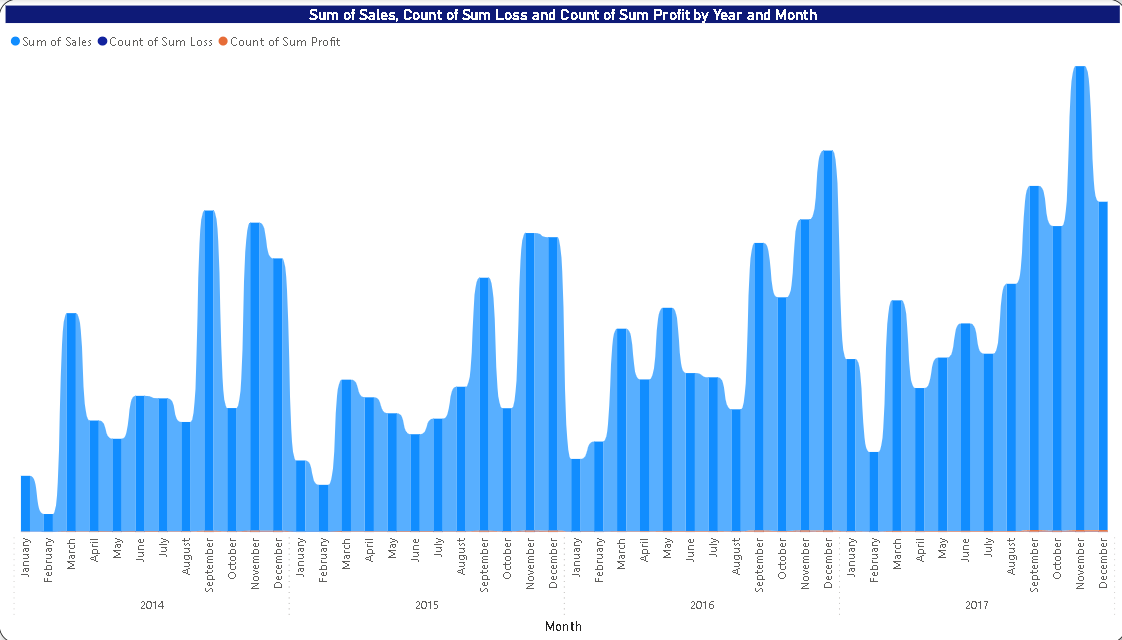


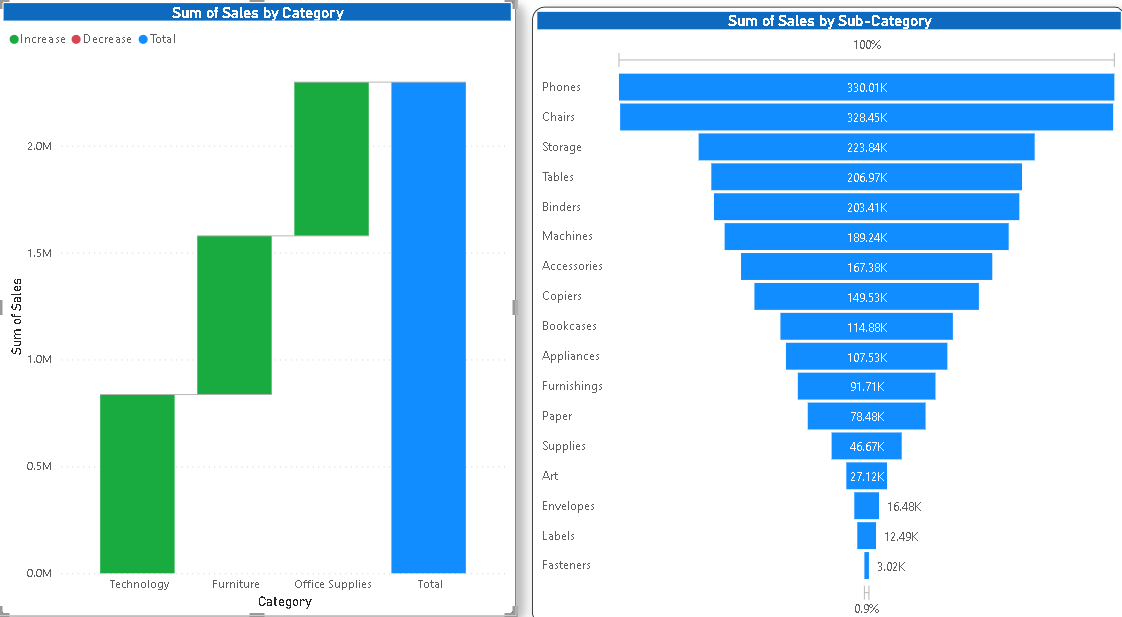
**Visualizing Data:**

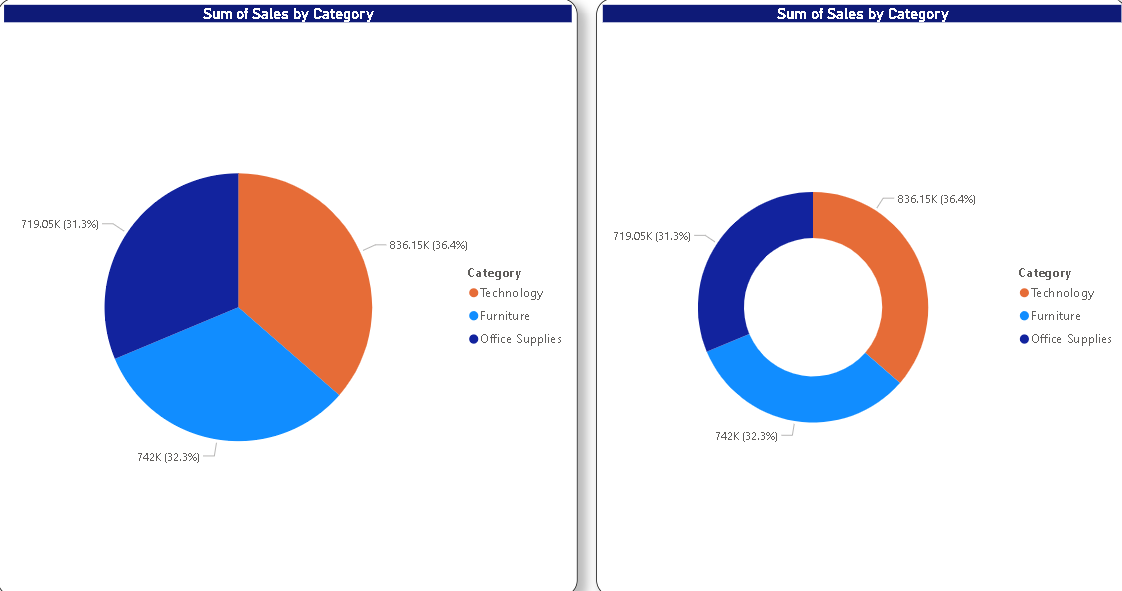
****

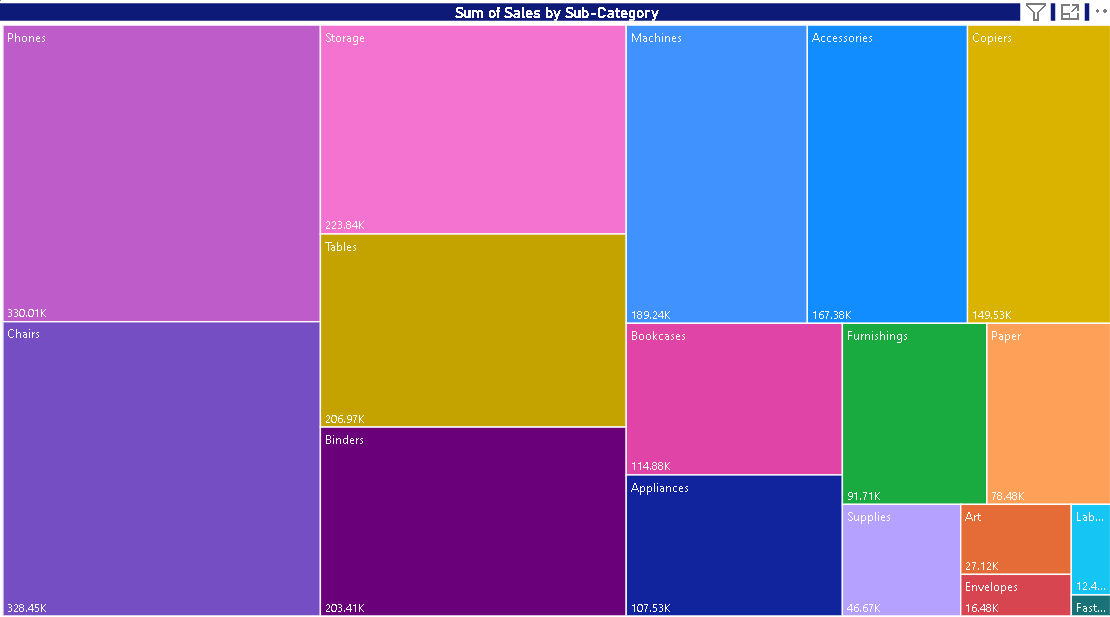


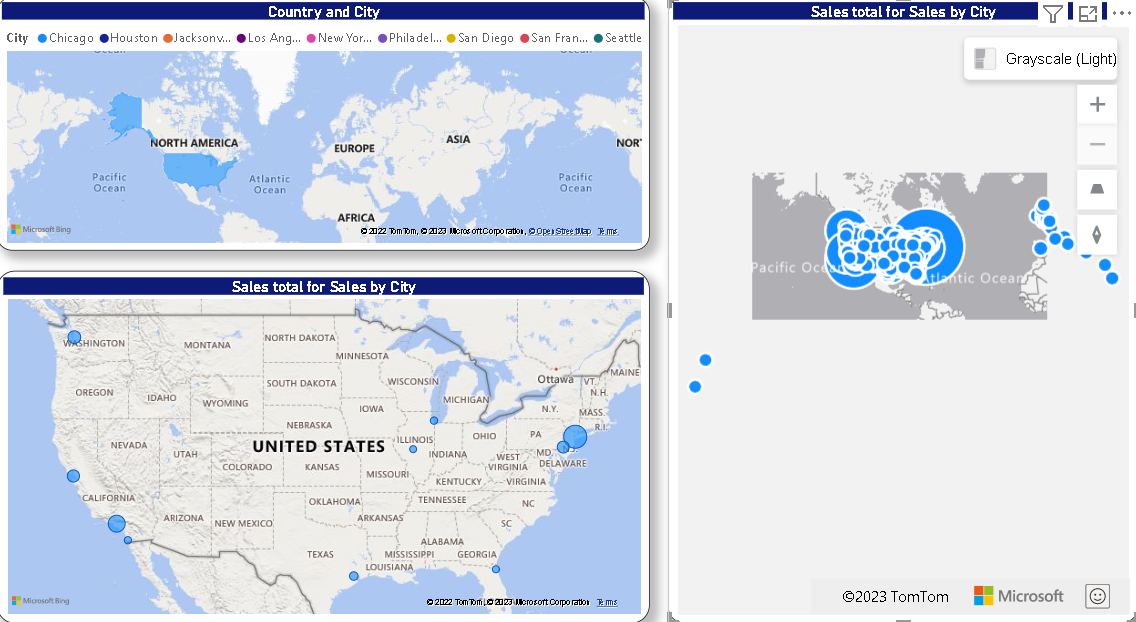


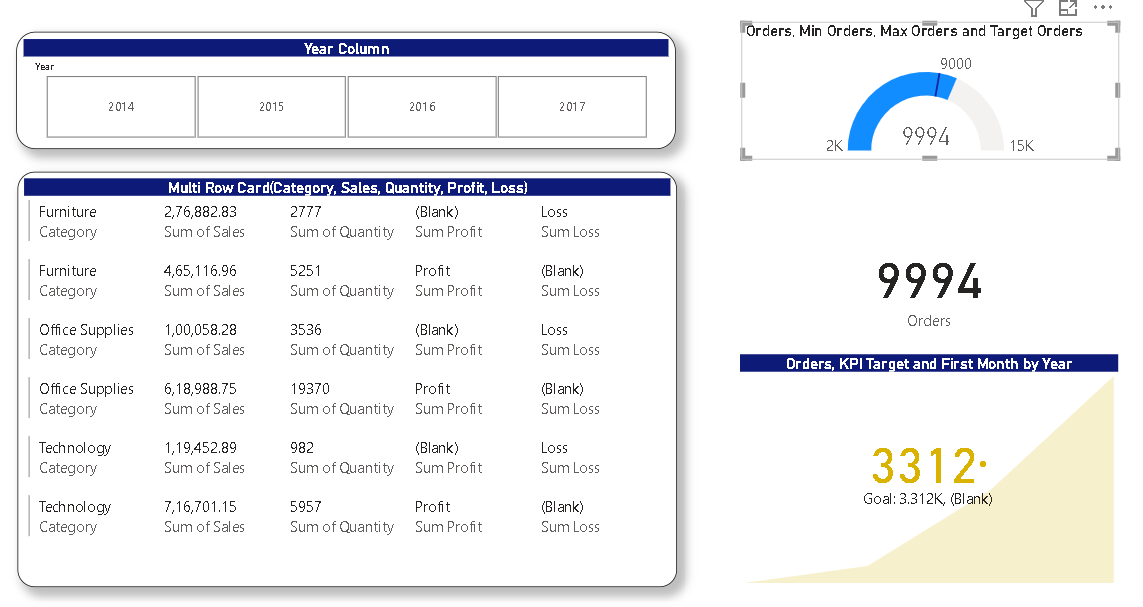


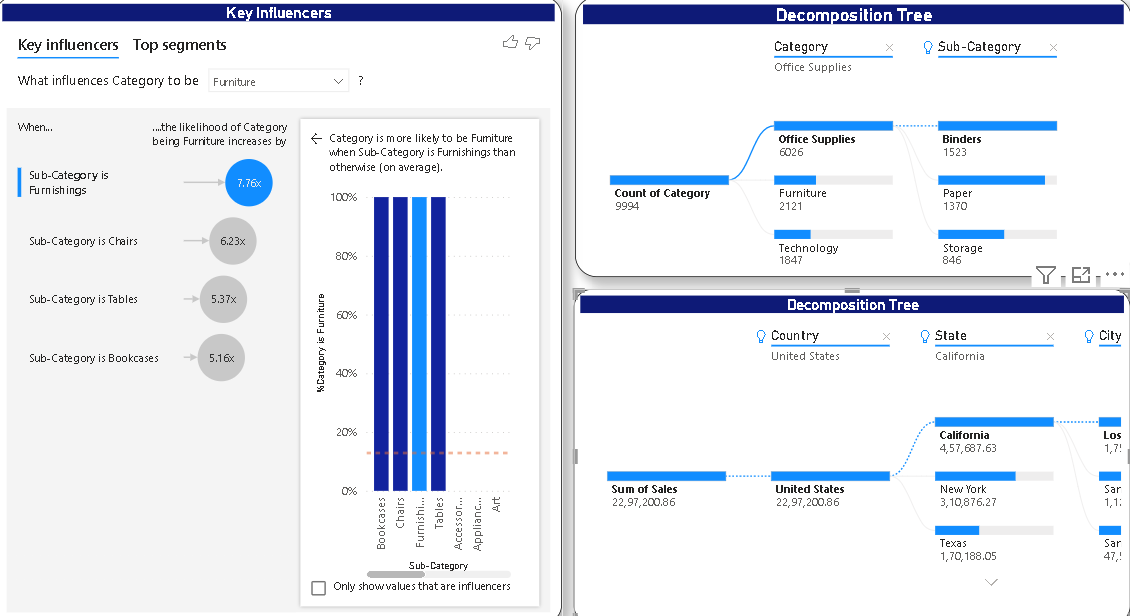












**Assignment-3**

Power BI provides a variety of built-in functions that you can use to manipulate and analyze your data. These functions can be categorized into different types based on their functionality. Here are the different types of functions available in Power BI:

1. Aggregation functions: These functions are used to perform calculations on a set of values and return a single result. Some common aggregation functions in Power BI include SUM, COUNT, AVERAGE, MAX, MIN, and DISTINCTCOUNT.
2. Date and Time functions: These functions are used to work with date and time values. Some common date and time functions in Power BI include YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, TODAY, NOW, and DATE.
3. Text functions: These functions are used to work with text values. Some common text functions in Power BI include CONCATENATE, LEFT, RIGHT, MID, LOWER, UPPER, PROPER, SUBSTITUTE, and TRIM.
4. Logical functions: These functions are used to perform logical operations on one or more values. Some common logical functions in Power BI include IF, AND, OR, NOT, TRUE, and FALSE.
5. Lookup functions: These functions are used to retrieve values from a table or a range of cells. Some common lookup functions in Power BI include VLOOKUP, HLOOKUP, INDEX, and MATCH.
6. Statistical functions: These functions are used to perform statistical analysis on a set of values. Some common statistical functions in Power BI include STDEV, AVERAGEIF, AVERAGEIFS, MEDIAN, MODE, PERCENTILE, and RANK.
7. Financial functions: These functions are used to perform financial calculations. Some common financial functions in Power BI include PV, FV, PMT, NPV, IRR, and RATE.
8. Information functions: These functions are used to retrieve information about a cell or a range of cells. Some common information functions in Power BI include CELL, ISBLANK, ISNUMBER, ISTEXT, and TYPE.
9. Math functions: These functions are used to perform mathematical operations on one or more values. Some common math functions in Power BI include ABS, ROUND, CEILING, FLOOR, EXP, LOG, and SQRT.
10. Conversion functions: These functions are used to convert values from one data type to another. Some common conversion functions in Power BI include VALUE, DATEVALUE, TIMEVALUE, and TEXT.
11. Time Intelligence functions: These functions are used to perform calculations on dates and times for time-based analysis. Some common time intelligence functions in Power BI include SAMEPERIODLASTYEAR, DATESYTD, TOTALYTD, and CLOSINGBALANCEYEAR.
12. Filter functions: These functions are used to filter data based on specific criteria. Some common filter functions in Power BI include FILTER, CALCULATETABLE, and ALL.
13. Ranking functions: These functions are used to rank values based on a specific criterion. Some common ranking functions in Power BI include RANKX, TOPN, and BOTTOMN.
14. Grouping functions: These functions are used to group data based on specific criteria. Some common grouping functions in Power BI include GROUPBY, SUMMARIZE, and ADDCOLUMNS.
15. Table functions: These functions are used to manipulate tables and create new tables. Some common table functions in Power BI include UNION, INTERSECT, EXCEPT, and NATURALINNERJOIN.
16. Time series functions: These functions are used to analyze time series data. Some common time series functions in Power BI include FIRSTDATE, LASTDATE, TOTALMTD, and TOTALQTD.
17. Geospatial functions: These functions are used to analyze data based on geographic location. Some common geospatial functions in Power BI include GEOGRAPHY, GEODISTANCE, and GEOHEXAGON.
18. Advanced functions: These functions are used for advanced data analysis and manipulation. Some common advanced functions in Power BI include GENERATE, CROSSJOIN, LOOKUPVALUE, and EVALUATE.
19. Custom functions: Power BI also allows you to create your own custom functions using the Power Query Editor or with M code. These functions can be used to perform specific data transformations or calculations that are not available with the built-in functions.

Overall, Power BI provides a wide range of functions to help you manipulate and analyze your data in a variety of ways. By understanding the different types of functions available and how to use them effectively, you can create powerful data models and visualizations to gain valuable insights from your data.

**An idea of how they work for some funtions:**

Top of Form

1. SUM: This function is used to add up a set of values. For example, you can use the SUM function to calculate the total sales for a particular product or category.
2. COUNT: This function is used to count the number of values in a set. For example, you can use the COUNT function to count the number of orders placed in a particular month.
3. IF: This function is used to perform a conditional test and return a specific value based on the result. For example, you can use the IF function to determine whether a salesperson met their sales quota and return "Met" or "Not Met" based on the result.
4. VLOOKUP: This function is used to look up a value in a table and return a related value from the same row. For example, you can use the VLOOKUP function to look up a product code in a table and return the corresponding product name.
5. STDEV: This function is used to calculate the standard deviation of a set of values. For example, you can use the STDEV function to calculate the standard deviation of a set of test scores.
6. PV: This function is used to calculate the present value of an investment. For example, you can use the PV function to calculate the present value of a loan or an annuity.
7. ISBLANK: This function is used to check whether a cell is blank and return TRUE or FALSE based on the result. For example, you can use the ISBLANK function to check whether a cell in a table contains a value or is empty.
8. ROUND: This function is used to round a number to a specified number of decimal places. For example, you can use the ROUND function to round a sales amount to the nearest dollar.
9. DATEVALUE: This function is used to convert a text string to a date value. For example, you can use the DATEVALUE function to convert a date in text format to a date value that can be used in calculations.
10. RANKX: This function is used to rank values in a table based on a specified criterion. For example, you can use the RANKX function to rank salespeople based on their total sales for a particular period.

These are just a few examples of some common functions in Power BI. The specific examples you use will depend on your data and your analysis goals.

Top of Form

**most commonly used tools in the Power BI ribbon:**

1. Home: This tab contains a variety of tools for creating and formatting visualizations, such as changing colors, fonts, and labels. You can also use the Home tab to manage your data connections and refresh your data.
2. Insert: This tab allows you to add new visualizations and other elements to your report, such as tables, charts, and images. You can also use the Insert tab to add new pages to your report or to add shapes and text boxes.
3. Modeling: This tab contains tools for managing your data model, such as creating new relationships between tables, adding calculated columns, and creating hierarchies. You can also use the Modeling tab to create and manage data measures and to create calculated tables.
4. View: This tab allows you to control the layout and appearance of your report, such as zooming in or out, switching between different views, and toggling between report and data views. You can also use the View tab to show or hide specific elements of your report, such as gridlines or visual headers.
5. Data: This tab contains tools for managing your data sources, such as importing new data, creating queries, and managing data transformations. You can also use the Data tab to view and edit the raw data that you are working with.
6. Transform Data: This tab allows you to perform data transformations and manipulations using Power Query, such as merging tables, filtering data, and removing duplicates. You can also use the Transform Data tab to create custom functions and to apply transformations to multiple tables at once.
7. Format: This tab contains tools for formatting your report elements, such as adjusting margins and padding, adding borders, and changing the alignment of text and images. You can also use the Format tab to add custom themes and backgrounds to your report.

Overall, the Power BI ribbon provides a wide range of tools for creating and customizing your reports and visualizations. By understanding the different tabs and tools available, you can create powerful and engaging reports that effectively communicate your data insights.

**Data transformation:**

1. Filtering: You can use filtering to remove unwanted rows or columns from your data. For example, you can filter out rows that contain missing or invalid data.
2. Sorting: You can use sorting to arrange your data in a specific order. For example, you can sort your data by date or by alphabetical order.
3. Grouping: You can use grouping to combine data based on certain criteria. For example, you can group your sales data by product category to see the total sales for each category.
4. Pivot: You can use pivoting to change the structure of your data. For example, you can pivot your sales data to show the total sales for each product category by month.
5. Splitting and merging columns: You can use splitting to separate a single column into multiple columns, and you can use merging to combine multiple columns into a single column.
6. Adding or removing columns: You can add or remove columns from your data to include or exclude specific information. For example, you can add a calculated column to calculate the total sales for each product.
7. Data type conversion: You can convert data types to ensure that your data is correctly interpreted by Power BI. For example, you can convert a text column to a date column.
8. Cleaning data: You can clean your data by removing duplicates, correcting errors, and formatting data consistently. For example, you can remove duplicate rows or correct misspelled names.
9. Combining data from multiple sources: You can combine data from multiple sources by merging or appending tables.

Overall, data transformation is a critical step in the data analysis process. By using Power Query Editor to transform your data, you can ensure that your data is accurate, complete, and ready for analysis.

Top of Form

**Power BI provides three main views for working with your data: Report view, Model view, and Data view:**

1. Report view: This is the default view in Power BI, and it is where you create and design your reports. In Report view, you can add and customize visualizations, create dashboards, and add text and images. You can also interact with your data by filtering, drilling down, and drilling up to see different levels of detail. Report view is designed to be highly interactive, allowing you to explore and analyze your data in real time.
2. Model view: Model view is where you manage the relationships between the tables in your data model. In Model view, you can create and edit relationships, add calculated columns and measures, and manage your data types. Model view is designed for data modeling experts, and it provides a more detailed and granular view of your data model than Report view.
3. Data view: Data view is where you manage your data sources and queries. In Data view, you can connect to new data sources, create and edit queries, and transform your data using Power Query Editor. Data view is designed for data analysts and data engineers, and it provides a powerful set of tools for working with your data.

Overall, these three views provide a comprehensive set of tools for working with your data in Power BI. By using each view appropriately, you can design effective reports, create accurate data models, and transform your data in meaningful ways.

Top of Form

**DATABASE**

**PostgreeSQL:**

PostgreSQL is an open-source relational database management system (RDBMS) that is designed to be highly scalable and customizable. It is often used as an alternative to commercial database systems such as Oracle, Microsoft SQL Server, and MySQL.

Some key features of PostgreSQL:

1. Open-source: PostgreSQL is released under the PostgreSQL License, which is a permissive open-source license that allows anyone to use, modify, and distribute the software.
2. ACID-compliant: PostgreSQL is ACID-compliant, which means that it supports transactions that are atomic, consistent, isolated, and durable.
3. Extensible: PostgreSQL is highly extensible, and it allows developers to create custom data types, operators, functions, and indexes.
4. Scalable: PostgreSQL is designed to be highly scalable, and it can support large databases with many users and complex queries.
5. Cross-platform: PostgreSQL is available for multiple platforms, including Windows, Linux, and macOS.
6. SQL-compliant: PostgreSQL supports the SQL standard, and it includes many advanced SQL features such as window functions, common table expressions, and recursive queries.
7. Security: PostgreSQL provides a wide range of security features, including SSL encryption, authentication and authorization mechanisms, and support for row-level security.
8. Replication: PostgreSQL includes built-in support for replication, which allows you to create replicas of your database for high availability and disaster recovery.
9. Support for JSON and other non-relational data types: PostgreSQL includes support for storing and querying JSON data, as well as other non-relational data types such as arrays and hstore.

Overall, PostgreSQL is a highly capable RDBMS that provides a wide range of advanced features for developers and database administrators. Its flexibility, scalability, and security features make it a popular choice for many applications, from small startups to large enterprises.

## Why Windward for SQL?

With Windward, you deliver better looking reports, on time.

The Windward product suite helps you embed advanced reporting and document generation functionality into your new or existing software application. The **Windward Java and .NET engines** quickly and cleanly embed into your codebase and connect with your SQL-based data source or sources.

By employing the Windward reporting and document generation solution, you to take advantage of SQL Server’s best features.

**The advantages include:**

### Spend Less Time Programming

The ability to**apply your existing queries, views and stored procedures** allows you to get started quickly. Views and stored procedures allow you to reuse the queries you have worked so hard to perfect. Using SQL language queries directly in Windward tags enables you to harness the power of the large library of built-in functions. Windward takes advantage of these time-saving SQL features natively, allowing you to focus on template creation, not data selection.

### Expand Your Data Selection Options

SQL’s functions provide you with a whole toolbox full of data selection options. **Windward’s ability to use these functions natively extends those options,**meaning lyou spend less time crunching your data and even more time using it.

### Utilize Powerful Joins

SQL’s joins are a great feature to link tables together by a common column. Building and perfecting them, however, can be a time-consuming process. **Windward’s SQL wizard has a unique algorithm that will automatically create joins between your tables.**

The algorithm scans your SQL Server’s metadata and distills those table relations for you. This is all bundled neatly in Windward’s SQL Wizard, automatically creating joins based on the data you select.

### Relax with Validated Security

You created a strong security layer for your SQL Server, so you should use a product that fully utilizes it.

**Windward connects and validates against your existing SQL Server security** via internal database or Active Directory. This enables you to design and deliver your reports securely. You'll have the peace of mind that your end users will only have access to the data that you allow them to.

### Create Great Templates with Ease

You complete your programming, non-developer employees and other end users apply **the Windward AutoTag design tool** to create report templates in Microsoft® Office®. Novice users can apply Windward’s intuitive SQL wizard to create complex queries with simple drag and drop actions, directly linking their templates to your SQL-based data source.

End users can even output those reports when and where they want, in widely used and widely accepted formats like DOCX, PDF, HTML, direct-to-printer, and more.

‍

## Key Benefits for Oracle Database Reporting with Windward:

* **A database reporting solution that runs on YOUR platform**. Windward is available for Java (pure Java), .NET (100% managed code) and almost any other programming language you may use.
* **You choose your programming language.**Create high-performance, multi-threaded and modular applications in Java, C#, VisualBasic, C++, Python, or PHP that you can build and run on desktop, server, and embedded platforms.
* **Easy to embed.**Incorporating Windward into your existing program takes as little as 14 lines of code. (Note: template design is in Office but the engine does not use Office.)
* **Generate complex queries easily.**The SQL Wizard is so intuitive that even someone new to PostgreSQL reporting can generate almost any query.
* **Streamlined data access.**Createdatasets of complex views to provide simple tables to access that data.
* **Easy database reporting with fast and flexible template design.**You design reports in Microsoft Word and Excel, so there’s little training or learning curve – and you get a UI that is superbly designed for document generation. Plus, the non-banded design tool lets you do things that are impossible with other Oracle reporting tools.
* **Significant time savings through reusable elements.**Windward Pods give you a powerful way to drag and drop data tags and complex sub reports – with no limitation on the combination of documents and tags in the Pod. Plus, you can pull in additional text and documents with a single tag, repeat parts of the document, include conditional parts of the document, and more.
* **Powerful data selection.**Commands to select data include a large number of equations and functions, and you can easily add your own for more power.
* **Speedy report generation.**Report generation is measured in pages per second, not seconds per page.

Top of Form

**CRUD Operations:**

CRUD stands for Create, Read, Update, and Delete, which are the four basic operations that can be performed on data in a database. Here's how each of these operations works in PostgreSQL:

1. Create: The CREATE operation allows you to create new records in a table. To create a new record in PostgreSQL, you would use the INSERT statement, which allows you to specify the values for each column in the table. Here's an example of how to create a new record in a table:

Example:

INSERT INTO my\_table (column1, column2, column3) VALUES ('value1', 'value2', 'value3');

This statement creates a new record in **my\_table** with the values **value1**, **value2**, and **value3** for the columns **column1**, **column2**, and **column3**, respectively.

1. Read: The READ operation allows you to retrieve records from a table. To read records from a table in PostgreSQL, you would use the SELECT statement, which allows you to specify which columns to retrieve and which conditions to filter on. Here's an example of how to read records from a table:

Example:

SELECT \* FROM my\_table WHERE column1 = 'value1';

This statement retrieves all records from **my\_table** where the value in **column1** is **value1**.

1. Update: The UPDATE operation allows you to modify existing records in a table. To update a record in PostgreSQL, you would use the UPDATE statement, which allows you to specify which columns to update and which conditions to filter on. Here's an example of how to update a record in a table:

Example:

UPDATE my\_table SET column1 = 'new\_value1' WHERE column2 = 'value2';

This statement updates the value in **column1** to **new\_value1** for all records in **my\_table** where the value in **column2** is **value2**.

1. Delete: The DELETE operation allows you to delete records from a table. To delete records from a table in PostgreSQL, you would use the DELETE statement, which allows you to specify which records to delete based on conditions. Here's an example of how to delete records from a table:

Example:

DELETE FROM my\_table WHERE column1 = 'value1';

This statement deletes all records from **my\_table** where the value in **column1** is **value1**.

Overall, CRUD operations are fundamental to working with databases, and PostgreSQL provides powerful SQL statements that make it easy to create, read, update, and delete records in a table.

Top of Form

**Example of how to create a table in PostgreSQL and perform CRUD operations on it:**

First, let's connect to a PostgreSQL database using the **psql** command-line tool:

**Script:**

psql -U postgres -d my\_database

Assuming you have created a database named **my\_database**, this command will connect you to it using the default user **postgres**.

Next, let's create a table called **users** with three columns: **id**, **name**, and **email**. Here's the SQL script to create the table:

**Script:**

CREATE TABLE users (

id SERIAL PRIMARY KEY,

name TEXT NOT NULL,

email TEXT UNIQUE NOT NULL

);

This script creates a table with three columns:

* **id** is a serial primary key column, which means that PostgreSQL will automatically generate a unique integer value for this column for each new record.
* **name** is a required text column, which means that it cannot be null.
* **email** is a unique required text column, which means that it cannot be null and that no two records can have the same value for this column.

Next, let's insert some data into the **users** table:

**Script:**

INSERT INTO users (name, email) VALUES ('John', 'john@example.com');

INSERT INTO users (name, email) VALUES ('Jane', 'jane@example.com');

These two statements create two new records in the **users** table, one for John and one for Jane.

To read the data from the **users** table, we can use the following SELECT statement:

**Script:**

SELECT \* FROM users;

This will return a result set with all the records in the **users** table:

**Script:**

**id | name | email**

**----+------+------------------**

**1 | John | john@example.com**

**2 | Jane | jane@example.com**

**(2 rows)**

To update a record, we can use the following UPDATE statement:

**Script:**

**UPDATE users SET name = 'Johnny' WHERE id = 1;**

This statement updates the **name** column for the record with **id** equal to 1 to **'Johnny'**.

To delete a record, we can use the following DELETE statement:

**Script:**

DELETE FROM users WHERE id = 2;

This statement deletes the record with **id** equal to 2 from the **users** table.

Overall, these four SQL statements (CREATE, INSERT, SELECT, UPDATE, and DELETE) provide the basic building blocks for performing CRUD operations on a PostgreSQL database.

**Example:**

To create a table to store information about students in a university. The table should have columns for the student's name, email address, major, enrollment year, and a unique student ID. We'll also include a foreign key reference to another table for the student's academic advisor.

**Script:**

CREATE TABLE students (

id SERIAL PRIMARY KEY,

name TEXT NOT NULL,

email TEXT UNIQUE NOT NULL,

major TEXT,

enrollment\_year INTEGER NOT NULL,

advisor\_id INTEGER REFERENCES advisors(id)

);

This script creates a table with six columns:

* **id** is a serial primary key column, which means that PostgreSQL will automatically generate a unique integer value for this column for each new record.
* **name** is a required text column, which means that it cannot be null.
* **email** is a unique required text column, which means that it cannot be null and that no two records can have the same value for this column.
* **major** is an optional text column.
* **enrollment\_year** is a required integer column, which means that it cannot be null.
* **advisor\_id** is an integer column that references the **id** column in the **advisors** table. This creates a foreign key constraint, which ensures that any value in the **advisor\_id** column must also exist in the **id** column of the **advisors** table.

Next, let's insert some data into the **students** table:

**Script:**

**INSERT INTO students (name, email, major, enrollment\_year, advisor\_id)**

**VALUES ('John Smith', 'jsmith@example.com', 'Computer Science', 2020, 1),**

**('Jane Doe', 'jdoe@example.com', 'Biology', 2019, 2),**

**('Bob Johnson', 'bjohnson@example.com', NULL, 2022, 1);**

These three statements create three new records in the **students** table.

To read the data from the **students** table, we can use the following SELECT statement:

**Script:**

SELECT \* FROM students;

This will return a result set with all the records in the **students** table:

id | name | email | major | enrollment\_year | advisor\_id

----+---------------+---------------------+---------------------+----------------+------------

1 | John Smith | jsmith@example.com | Computer Science | 2020 | 1

2 | Jane Doe | jdoe@example.com | Biology | 2019 | 2

3 | Bob Johnson | bjohnson@example.com| NULL | 2022 | 1

(3 rows)

To update a record, we can use the following UPDATE statement:

UPDATE students SET major = 'Electrical Engineering' WHERE id = 1;

This statement updates the **major** column for the record with **id** equal to 1 to **'Electrical Engineering'**.

To delete a record, we can use the following DELETE statement:

DELETE FROM students WHERE id = 2;

This statement deletes the record with **id** equal to 2 from the **students** table.

**DBeaver:**

DBeaver is a free, open-source database management tool that allows users to manage and manipulate data across multiple database systems. It supports a wide range of databases, including MySQL, PostgreSQL, Oracle, SQLite, and SQL Server, among others. DBeaver provides a graphical user interface (GUI) that allows users to easily create, edit, and execute SQL queries and perform other database-related tasks.

Here are some key features of DBeaver:

1. Multi-platform support: DBeaver is available for Windows, macOS, and Linux, making it accessible to a wide range of users.
2. Multiple database support: DBeaver supports a wide range of databases, including MySQL, PostgreSQL, Oracle, SQLite, and SQL Server, among others.
3. SQL Editor: DBeaver provides a powerful SQL editor that supports syntax highlighting, code completion, and code formatting.
4. Data Viewer: DBeaver allows users to view and edit data in a table view or form view.
5. Schema Editor: DBeaver provides a schema editor that allows users to create and modify database schema, including tables, views, and indexes.
6. Data Transfer: DBeaver allows users to transfer data between databases and CSV files.
7. Backup and Restore: DBeaver provides backup and restore functionality, allowing users to create and restore database backups.
8. Plugins: DBeaver supports a plugin architecture that allows users to extend the functionality of the tool with custom plugins.

Overall, DBeaver is a powerful and flexible database management tool that offers a wide range of features for managing and manipulating data across multiple database systems. Its support for multiple databases, powerful SQL editor, and other features make it a popular choice for developers and database administrators alike.

**DBeaver CRUD Operations:**

DBeaver provides a graphical user interface (GUI) that allows users to perform CRUD (Create, Read, Update, Delete) operations on databases. Here are some examples of how to perform CRUD operations using DBeaver:

1. Create operation: To create a new record in a table, you can use the INSERT statement. In DBeaver, you can right-click on a table in the Object Explorer and select "Insert Rows". This will open a new editor window where you can enter the data for the new record. Once you have entered the data, you can click "Execute" to insert the new record into the table.
2. Read operation: To retrieve data from a table, you can use the SELECT statement. In DBeaver, you can right-click on a table in the Object Explorer and select "Select Rows". This will open a new editor window where you can enter a query to retrieve the data you want. Once you have entered the query, you can click "Execute" to retrieve the data from the table.
3. Update operation: To update an existing record in a table, you can use the UPDATE statement. In DBeaver, you can right-click on a table in the Object Explorer and select "Edit Rows". This will open a new editor window where you can edit the data for the record you want to update. Once you have made your changes, you can click "Execute" to update the record in the table.
4. Delete operation: To delete a record from a table, you can use the DELETE statement. In DBeaver, you can right-click on a table in the Object Explorer and select "Delete Rows". This will open a new editor window where you can enter a query to delete the record(s) you want to remove. Once you have entered the query, you can click "Execute" to delete the record(s) from the table.

DBeaver also provides a wide range of other features that can be used to perform CRUD operations more efficiently. For example, you can use the Data Viewer to browse and edit data in a table, or use the Query Builder to create complex queries without having to write SQL code manually. Overall, DBeaver is a powerful and flexible database management tool that offers a wide range of features for performing CRUD operations and other database-related tasks.

Top of Form

**some examples of how to use DBeaver to perform common database tasks:**

1. Connecting to a database: To connect to a database using DBeaver, you first need to create a new database connection. To do this, click on the "New Connection" button in the toolbar and select the database type you want to connect to (e.g., PostgreSQL, MySQL, Oracle, etc.). Then, enter the connection details, such as the host name, port number, and credentials, and test the connection to make sure it works.
2. Creating a table: To create a new table in a database using DBeaver, you can right-click on the database in the Object Explorer and select "Create new Table". This will open a new editor window where you can specify the table name, column names, data types, and any constraints or indexes you want to apply. Once you have defined the table structure, you can click "Execute" to create the table in the database.
3. Querying data: To retrieve data from a table using DBeaver, you can use the SQL Editor to write and execute queries. For example, you can write a SELECT statement to retrieve all rows from a table, or use filters and sorting options to retrieve specific rows or columns. You can also use the Data Viewer to browse and edit data directly, without having to write SQL queries.
4. Modifying data: To modify data in a table using DBeaver, you can use the SQL Editor or the Data Viewer. For example, you can write an UPDATE statement to change the value of a specific column in a table, or use the Data Viewer to edit the data directly in the table. You can also use the Query Builder to create complex queries that involve multiple tables or conditions.
5. Exporting data: To export data from a table using DBeaver, you can use the Data Transfer feature. This allows you to select the table(s) you want to export, choose the file format (e.g., CSV, Excel, JSON, etc.), and specify the output location. You can also choose to export only specific columns or rows, or apply filters and sorting options to the data before exporting.

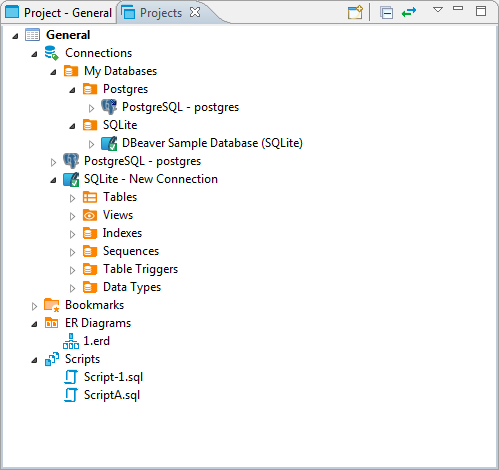
Overall, DBeaver provides a wide range of features and tools that make it easy to perform database-related tasks, from creating tables and querying data to modifying records and exporting data. Whether you are a database administrator, developer, or data analyst, DBeaver can help you work more efficiently and effectively with your databases.

Top of Form

# Projects View

You might need to classify and group database connections into projects. Projects store objects related not to a particular database but to all database connections. These are usually files stored on the file system.

The Projects view displays all projects created in the system and provides tools to manage them. To open the Projects view, on the **Window** menu, click **Projects** (or use ALT+W+P shortcut).



For information on how to change the view layout, please see the [Application Window Overview](https://dbeaver.com/docs/wiki/Projects-View/Application-Window-Overview) article.

The projects are organized into a tree and all have the same high-level structure:

* **Connections** – repeat the content of the Database Navigator view for this project. You can perform the same actions over the objects of the databases as in the Database Navigator.
* **Bookmarks** – contains bookmarks – shortcuts to database objects, see …
* **ER Diagrams** - contains ER diagrams that you can drag-and-drop here from other folders
* **Scripts** – contains scripts that you can drag-and-drop here from other folders

The Projects view provides a toolbar and View menu which contain generic items. Each object in the tree has its own context menu.

To open the view menu of the Projects view, click the View Menu button () in the upper-right corner of the window. The view menu contains the following items:

| **Icon** | **Item** | **Description** |
| --- | --- | --- |
|  | Create Project | Opens the Create Project wizard |
|  | Refresh Projects | Refreshes the projects tree to display changes caused by creating modifying or deleting projects |
|  | Collapse All | Collapses the tree to the root level |
|  | Link with editor | - Enabled when at least one editor is open, otherwise disabled - Highlights the object in the tree that has its editor open |

The toolbar is located in the title bar of the window, its buttons duplicate the view menu items except for the **Refresh Projects** one.

To open the context menu for an object in the tree, right-click the object. For information about context menu items of all objects under the **Connections** node of the tree, please see [Database Navigator](https://dbeaver.com/docs/wiki/Projects-View/Database-Navigator). The context menus of other nodes in the tree contain some basic items for copy-pasting, renaming, deleting objects, managing their properties, creating folders, etc.

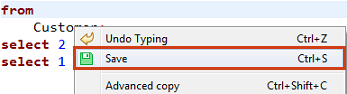
* The **Set Active Project** menu item (for a project root node) makes the project active, that is visible in the Database Navigator.
* The **Link File (SQL Script)** and **Link Folder** menu items allow creating links to files and folders in the file system.

# Script Management

## [Saving Scripts](https://dbeaver.com/docs/wiki/Script-Management/" \l "saving-scripts)

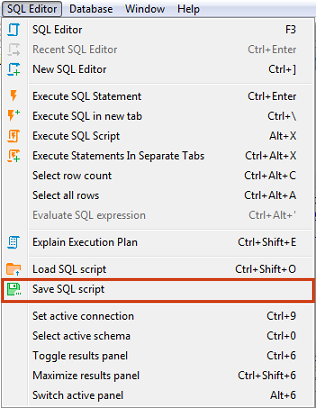
You can save scripts to a predefined space in the currently active project or somewhere in the file system.

To save a script to the current project space, just press Ctrl+S or right-click the script and click **Save** on the context menu:

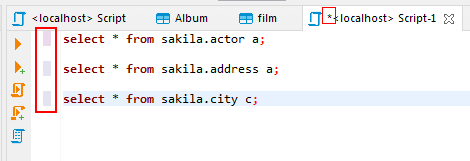


You can find the script saved this way in the [Project Explorer](https://dbeaver.com/docs/wiki/Script-Management/Project-Explorer) view in the **Scripts** folder.

To save a script to the file system, right-click the script, click **File -> Export SQL script** on the context menu and then select the folder in the file system. You can also click **SQL Editor -> Export SQL script** on the main menu:

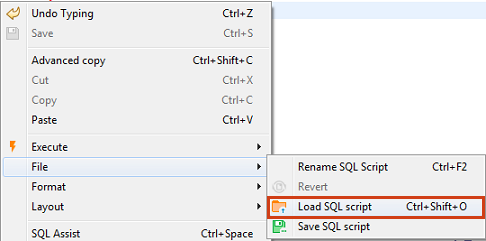


Unsaved data is highlighted in color on the left side of the editor, in addition to having an asterisk in the name of the script.



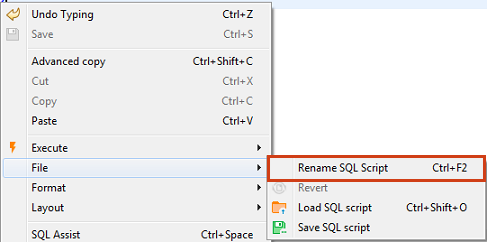
## [Loading Scripts](https://dbeaver.com/docs/wiki/Script-Management/" \l "loading-scripts)

To load a script stored in the file system to the SQL Editor, press CTRL+SHIFT+O, or click **SQL Editor -> Import SQL script** on the main menu, or right-click the script panel and click **File -> Import SQL script** on the context menu:

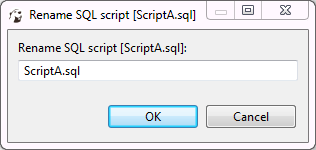


## [Renaming Scripts](https://dbeaver.com/docs/wiki/Script-Management/" \l "renaming-scripts)

To rename a script, right-click anywhere in the script panel, click **File -> Rename SQL Script** on the context menu or press CTRL+F2:



Then enter the new name in the Rename SQL script dialog box and click **OK**:

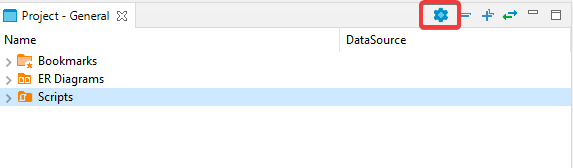
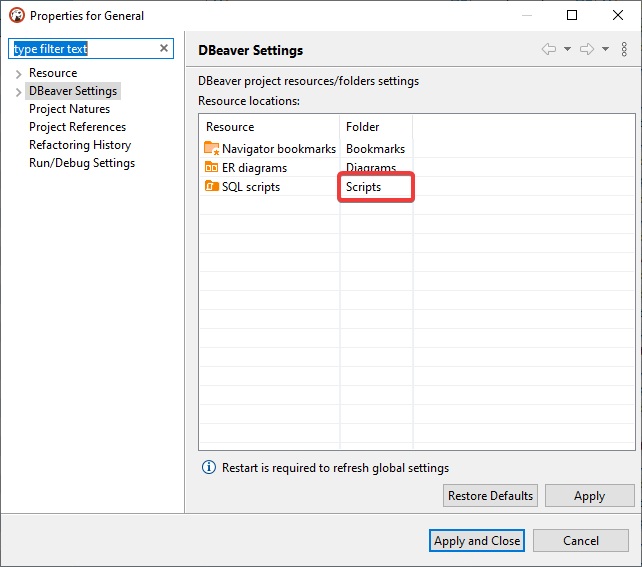


## [Reverting Changes](https://dbeaver.com/docs/wiki/Script-Management/" \l "reverting-changes)

If you want to revert all changes made to the current SQL script and return it to its initial state (reload from disk), right-click anywhere in the script panel and click **File -> Revert** on the context menu.

## [Changing default scripts directory](https://dbeaver.com/docs/wiki/Script-Management/" \l "changing-default-scripts-directory)

By default all of the scripts are saved to a "Scripts" folder located in your project inside the workspace directory. This can be changed by clicking the **Configure** button in [Project Explorer](https://dbeaver.com/docs/wiki/Script-Management/Project-Explorer) view. There you can click on a folder's name an pick any other folder inside the Project.

## [Adding external directory](https://dbeaver.com/docs/wiki/Script-Management/" \l "adding-external-directory)

You can also link an external directory to your project to either save your scripts into it, or to access scripts that were created outside of DBeaver.  
To link an External directory right-click anywhere in the Project Explorer and pick **Create -> Link Folder**. There you can link any directory on your drive to a project. This will allow you to open any externally created scripts through Project explorer and to set this folder as default to save new scripts into.

## [SQL Console](https://dbeaver.com/docs/wiki/Script-Management/" \l "sql-console)

In some cases you might want to execute a query and not save it in a script. For example, when you read table data using "Read data in SQL console" or open procedure/function source from DDL editor. SQL console does not have an associated .sql file. Its contents will be lost when you close it.

# Command Line

# [Command line parameters](https://dbeaver.com/docs/wiki/Command-Line/" \l "command-line-parameters)

Command line parameters might be passed directly to dbeaver[.exe] executable.  
In Windows, you can use dbeaver-cli.exe executable (it does not spawn a new window so you can see the output messages).

Also, you can add parameters in the dbeaver.ini configuration file. You need to write them to the beginning of the file, and each parameter has to be on its line.

## [DBeaver control](https://dbeaver.com/docs/wiki/Command-Line/" \l "dbeaver-control)

| **Name** | **Value** | **Example** |
| --- | --- | --- |
| -help | Prints help message |  |
| -stop | Quits DBeaver |  |
| -dump | Prints DBeaver thread dump |  |
| -f | Opens the file in DBeaver UI, if the command has -con argument, connects it to datasource | -f c:\some-path\some-file.sql |
| -con | Opens database connection in DBeaver UI | See [connection parameters table](https://dbeaver.com/docs/wiki/Command-Line/#connection-parameters) |
| -closeTabs | Closes all open editor tabs |  |
| -disconnectAll | Closes all open connections |  |
| -reuseWorkspace | Forces reuse of single workspace by multiple DBeaver instances |  |
| -newInstance | Forces new DBeaver instance creation (do not try to reuse already running one) |  |
| -bringToFront | Brings the DBeaver window on top of other applications |  |
| -var | Customs variables for runTask. You can change existing variables in the task. You cannot add new task variables with this parameter. You can add several parameters at once to the command line, each starting with "-var". Used right before -runTask. Template: -var variableName=variableValue | -var film=sakila.film -var actor=sakila.actor -runTask "exportFromSakila" EE version only. |
| -vars | Path to a property file with variables | -vars c:\path\to\file.properties For more information see [the main article](https://dbeaver.com/docs/wiki/Command-Line/Admin-Variables#declare-external-variables-in-a-file) |
| -runTask | Executes specified task | -runTask "@projectName:taskName". EE version only. See [task scheduler](https://dbeaver.com/docs/wiki/Command-Line/Task-Scheduler). |
| -license | Path to the EE license file | -license "/etc/licenses/dbeaver.txt". EE version only. |

## [System parameters](https://dbeaver.com/docs/wiki/Command-Line/" \l "system-parameters)

| **Name** | **Value** | **Example** |
| --- | --- | --- |
| -nl | Locale | en\_US |
| -data | Workspace path | c:\ProgramData\MyWorkspace |
| -nosplash | Omits splash screen | true |
| -clean | Clears all Eclipse caches. Use it if DBeaver fails to start after it upgrades. |  |
| -vmargs | VM parameters | See [VM arguments table](https://dbeaver.com/docs/wiki/Command-Line/#vm-arguments) |

### [VM arguments](https://dbeaver.com/docs/wiki/Command-Line/" \l "vm-arguments)

You can pass any advanced Java parameters supported by your local JVM (Oracle, OpenJDK, IBM, etc).  
Parameters supported by Oracle JVM (11): https://docs.oracle.com/en/java/javase/11/tools/java.html

Parameters supported by all JVMs:

| **Name** | **Value** | **Example** |
| --- | --- | --- |
| -Xms | Sets initial memory available for DBeaver | -Xmx1000m |
| -Xmx | Sets maximum memory available for DBeaver | -Xmx4000m |

### [Connection parameters](https://dbeaver.com/docs/wiki/Command-Line/" \l "connection-parameters)

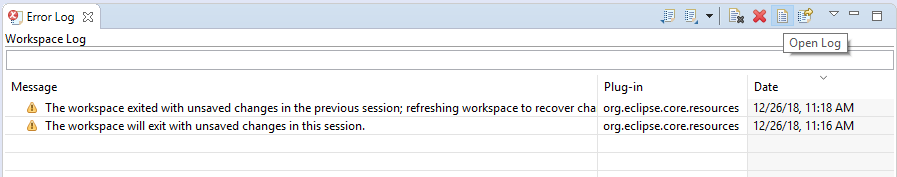
All connection parameters must be supplied as a single command line argument. The parameters are divided by pipe (|). The parameter name and value is divided by =.  
Example: -con "driver=sqlite|database=C:\db\SQLite\Chinook.db|name=SQLiteChin|openConsole=true|folder=SQLite"

| **Name** | **Description** | **Example** |
| --- | --- | --- |
| name | Connection name | Test connection |
| driver | Driver name or ID | driver=sqlite, driver=mysql, etc |
| url | Connection URL. Optional (JDBC URL may be constructed by a driver from other parameters) | url=jdbc:sqlite:C:\db\SQLite\Chinook.db |
| host | Database host name (optional) | host=localhost |
| port | Database port number (optional) | port=1534 |
| server | Database server name (optional) | server=myserver |
| database | Database name or path (optional) | database=db-name |
| user | User name (optional) | user=root |
| password | User password (optional) | password=mysecret |
| auth | Authentication model ID. See [Auth models](https://dbeaver.com/docs/wiki/Command-Line/Database-authentication-models) | auth=postgres\_pgpass |
| authProp.propName | Custom authentication parameters (depends on the driver and [auth model](https://dbeaver.com/docs/wiki/Command-Line/Database-authentication-models)) | authProp.oracle.net.wallet\_location=C:/temp/ora-wallet |
| savePassword | Does not ask user for a password on connection | savePassword=true |
| showSystemObjects | Shows/Hides system schemas, tables ,etc | showSystemObjects=true |
| showUtilityObjects | Shows/Hides utility schemas, tables ,etc | showUtilityObjects=true |
| folder | Puts a new connection in a folder | folder=FolderName |
| autoCommit | Sets connection auto commit flag (default value depends on driver) | autoCommit=true |
| prop.propName | Advanced connection parameters (depend on driver) | prop.connectTimeout=30 |
| id | Connection id | oracle\_thin-16a88e815bd-70598e648cedd28c (useful in conjunction with create=false) |
| connect | Connects to this database | connect=false |
| openConsole | Opens the SQL console for this database (sets connect to true) | openConsole=true |
| create | Creates new connection | create=false (true by default). If it is set as false, then an existing connection configuration will be used. The name or id parameter must be specified. |
| save | Saves new connection | When create=true, then save=false (default) makes new connection temporary, save=true means that new connection will be saved and accessible between DBeaver launches. |

# Log files

### [Error Log view](https://dbeaver.com/docs/wiki/Log-files/" \l "error-log-view)

There is an Error Log view (main menu Window->Show View->Error Log) which contains all errors which occur during the DBeaver runtime.  
You can double click on the warning/error in the log viewer and see the error stacktrace. Please attach it to the bug report.  
Also, you can open the full log (all error messages) if you need:



### [Log files](https://dbeaver.com/docs/wiki/Log-files/" \l "log-files)

DBeaver writes different log files. Most of them are Eclipse logs.  
Log files usually reside in the [workspace](https://dbeaver.com/docs/wiki/Log-files/Workspace-Location)/workspace6/.metadata .

* In Windows open Explorer and paste path %APPDATA%\DBeaverData\workspace6\.metadata.
* In Linux just type cd $XDG\_DATA\_HOME/DBeaverData/workspace6/.metadata
* In MacOS open path ~/Library/DBeaverData/workspace6/.metadata in Finder.
  + To view hidden folders press Cmd+Shift+. in the folder view.

Two standard log files:

* [workspace](https://dbeaver.com/docs/wiki/Log-files/Workspace-Location)/workspace6/.metadata/.log - all warnings and errors which happen during normal work
* [workspace](https://dbeaver.com/docs/wiki/Log-files/Workspace-Location)/workspace6/.metadata/dbeaver-debug.log - the same as .log plus debug information

In special cases log files can be written in other directories. A special case is an emergency situation when DBeaver cannot start and there is no workspace. Two typical places to find emergency logs:

* <install-path>/configuration
* ${HOME}/.eclipse/org.jkiss.dbeaver.product\_<dbeaver-version>

If you are reporting an error, please attach the applicable part of the log - not the complete file.  
Logs are very useful. Many errors cannot be reproduced and fixed without a full error stacktrace (all the details).

### [Java fatal logs](https://dbeaver.com/docs/wiki/Log-files/" \l "java-fatal-logs)

On the rare occasion that the DBeaver process dies, it does not leave any valuable logs. This is caused by a Java VM crash.  
JVM creates a fatal log file for each crash (log gile hs\_err\_PID.log). This log usually resides in the same directory where the DBeaver launcher is (e.g. dbeaver.exe).  
But in some cases it is a write-protected directory and the log file will be created in other folder.

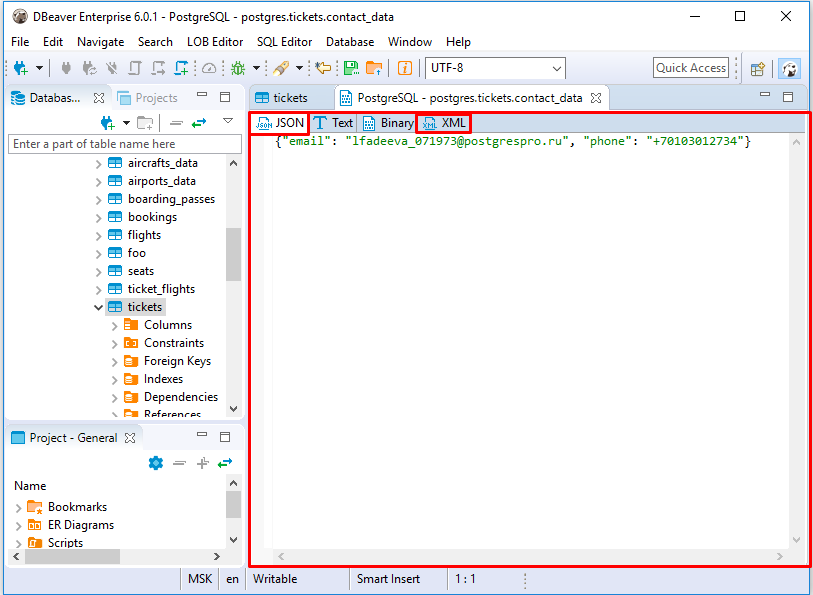
# Working with XML and JSON

DBeaver supports XML and JSON column types (in relational databases) by using standard JDBC interfaces. This feature was added in JDBC4 so you will need JDBC4 compliant driver for your database.

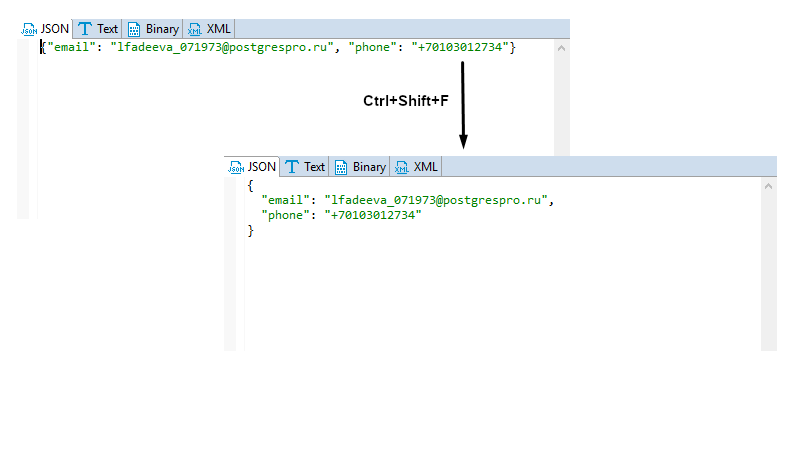
In the Data Editor, you can edit XML/JSON data right in the table cells/ However, a huge amount of data may require a larger editor so you might want to save XML/JSON scripts to a local file or upload this type of data from a local file.

To open the full-size XML/JSON editor click the cell containing data in XML/JSON format and press Shift+Enter.

By default the editor opens on JSON tab, open XML tab to modify XML data.

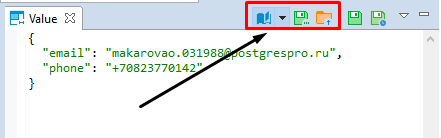


To auto-format XML/JSON script press Ctrl+Shift+F keyboard buttons.



Use Ctrl+S keyboard shortcut to save the changes made.

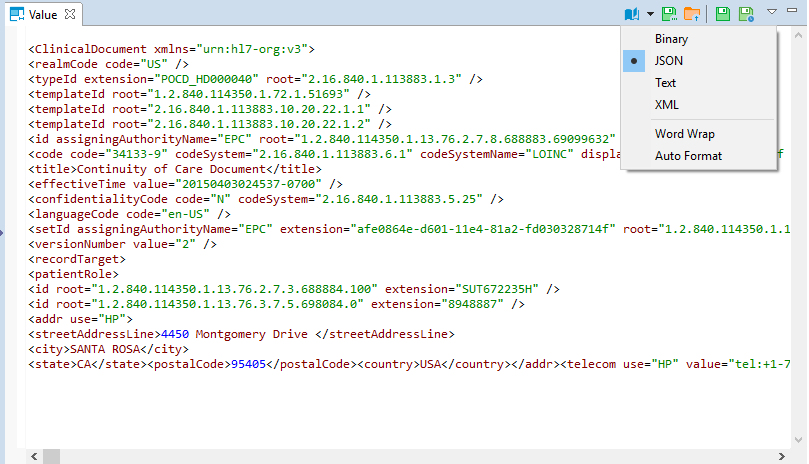
You can also edit XML/JSON content, save it locally and upload it from a local file with the help of **Value** panel toolbar.



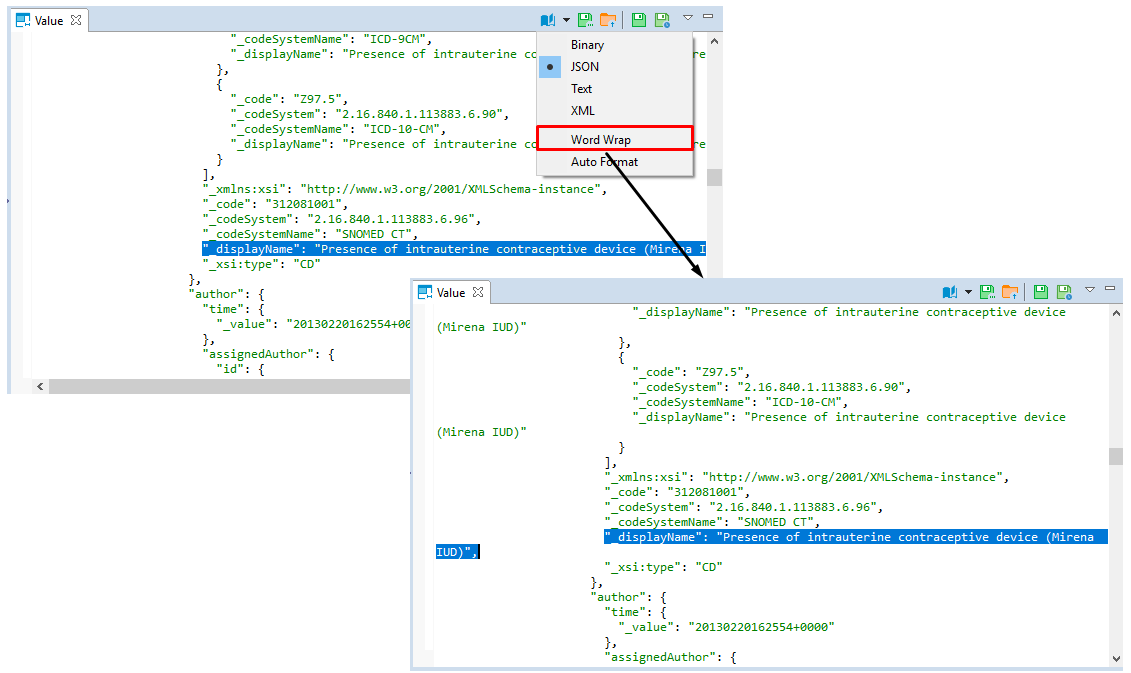
To upload data from a local file, press the **Load from file...** button .

To save the content to a local file, press the **Save to file...** button .

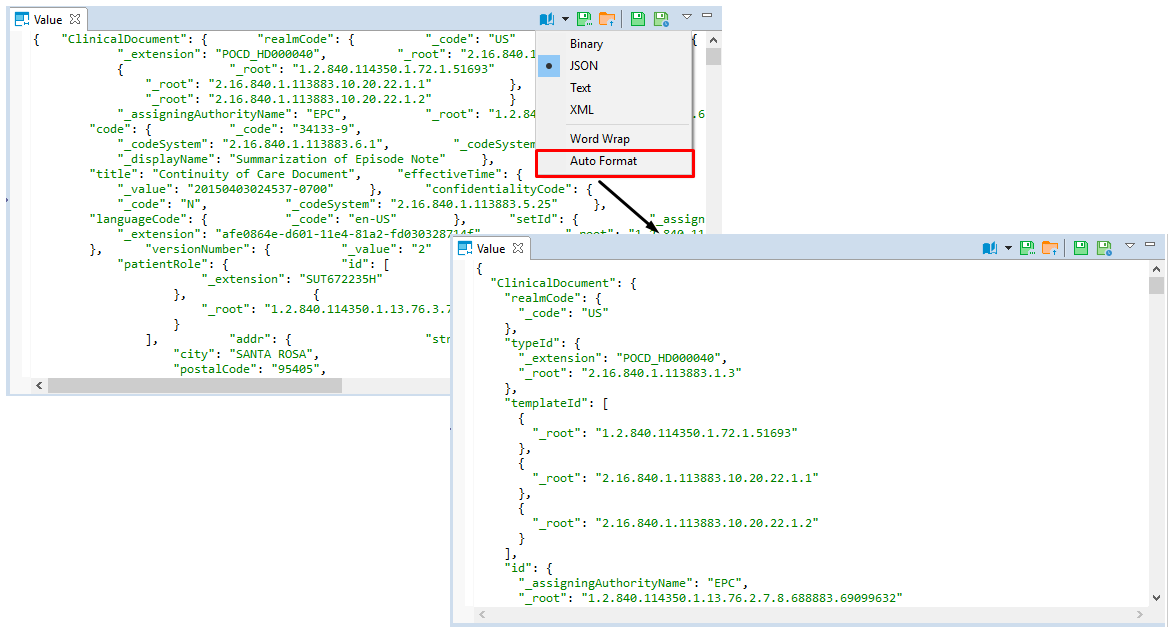
To switch between the formats, press the **Content viewer settings** button  and select the format.



Use **Word Wrap** feature that wraps the text within a screen.



Use **Auto Format** feature to automatically change the appearance of XML/JSON script (fix spaces around operators / commas, fix indentation, etc) and make it more readable.



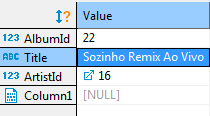
# Data View and Format

## [Grid vs. Plain Text Views](https://dbeaver.com/docs/wiki/Data-View-and-Format/" \l "grid-vs.-plain-text-views)

You can switch between two data presentations in SE version and four presentations in EE version. Pressing CTRL+~ switches available presentations in turn.

* To see the data in a grid view, similar to an Excel spreadsheet, click the **Grid** button () on the bottom toolbar of the editor.
* To switch to the plain text view, click **Text** () on the bottom toolbar.
* To switch to JSON view (available in EE version only for MongoDB documents and JSON tables), click **JSON** on the toolbar.
* To switch to XML view (available in EE version only for XML tables), click **XML** on the toolbar.

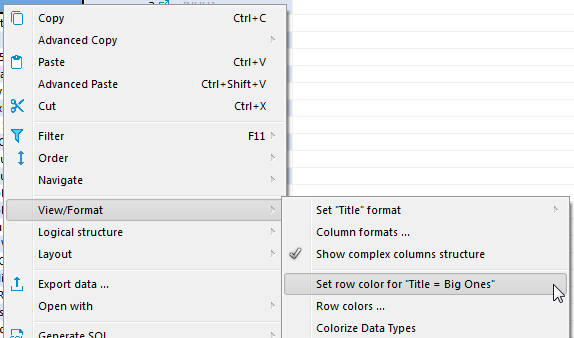
## [Table vs. Record Views](https://dbeaver.com/docs/wiki/Data-View-and-Format/" \l "table-vs.-record-views)

The table view is a standard table (Excel-like) in which columns are vertical and rows are horizontal. This view is the default one. If you click the **Record** button in the bottom toolbar of the editor (), or press Tab, or right-click a cell and then click **Layout -> Record** on the context menu, the rows and columns switch positions. The columns will appear as rows, and the rows will be hidden in one **Value** column which will show only one row of data. The column headers will shift from the top of the table to its left side:  
  
The Record view is useful if the table contains a big number of columns. To navigate from row to row of data, use the navigation buttons on the bottom toolbar of the editor: 

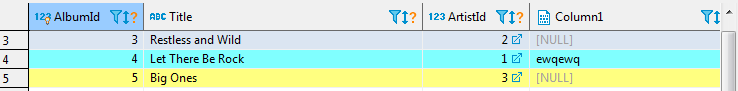
To return back to the standard table view, click the **Record** button again.

## [Rows Coloring](https://dbeaver.com/docs/wiki/Data-View-and-Format/" \l "rows-coloring)

In the data editor, you can colour all rows that have the same value as a particular cell of a certain column. To do so, right-click the cell and click View/Format ⇒ Set the row colour for {column name = value} on the context menu:

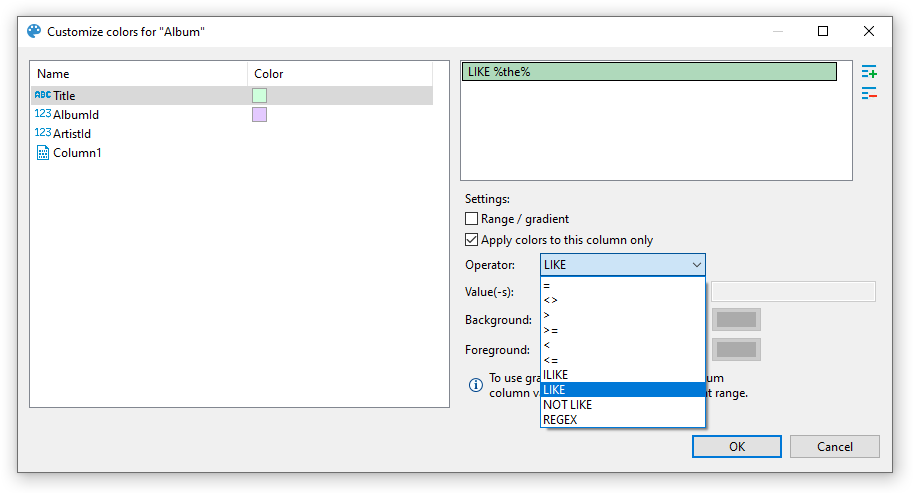


Then choose the colour in the palette window that appears and click **OK**. The current row and all other rows that contain the same value change their colour to the one you have selected:



To remove the coloring by a particular column, right-click the cell again and click View/Format ⇒ Clear colour for {column name = value} on the context menu.

By choosing View/Format ⇒ Row colors ... from the context menu, you can gain more precise control of coloring conditions:

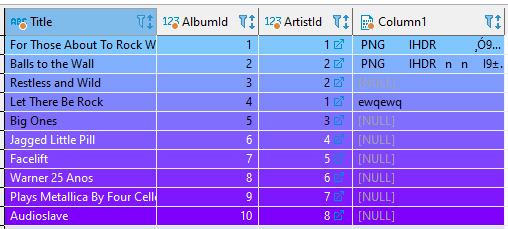


Here, you can define multiple conditions for single column using rich set of predefined operators, change background color and define a range between two values.

Operators work as you may expect. Note that they're executed on the client-side, that means no extra queries are made in order to apply colors.

### [Value range / Gradient](https://dbeaver.com/docs/wiki/Data-View-and-Format/" \l "value-range---gradient)

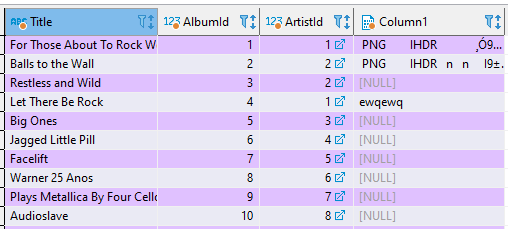
Value range allows you to paint your rows with gradient that fades from first value to second value:



In this example we defined a range for column AlbumId that fades from #80c6ff to #8000ff between values 1 and 10.

### [Using regex](https://dbeaver.com/docs/wiki/Data-View-and-Format/" \l "using-regex)

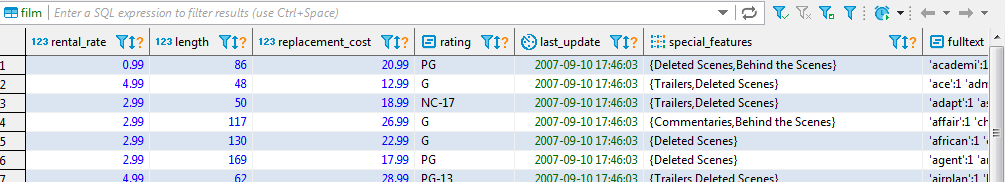
You can use regular expressions for matching complex values. Otherwise, you can be artistic and, for example, paint rows with odd values in your favorite color:



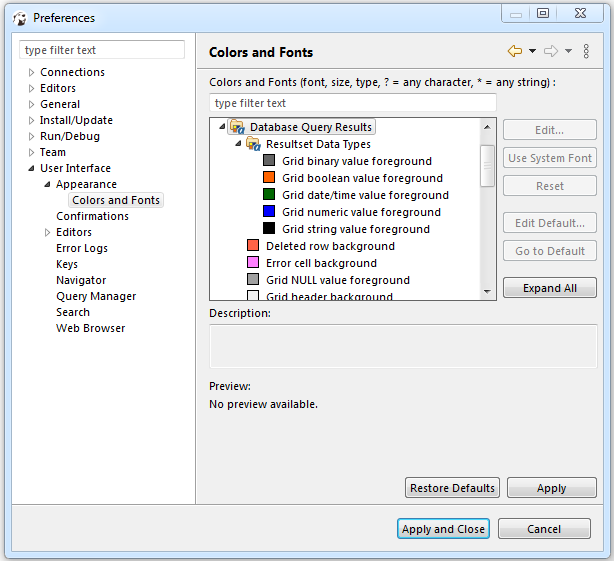
Snippet for coloring odd rows: *^\d\*[13579]$*; even rows: *^\d\*[02468]$*

## [Coloring by Data Types](https://dbeaver.com/docs/wiki/Data-View-and-Format/" \l "coloring-by-data-types)

Besides colouring rows by a value, you can colour the values in the columns by data types. To do so, right-click any cell in the table and, on the context menu, click **View/Format -> Colourize Data Types**. The values in the cells will be coloured in different colours according to the current colour preferences:



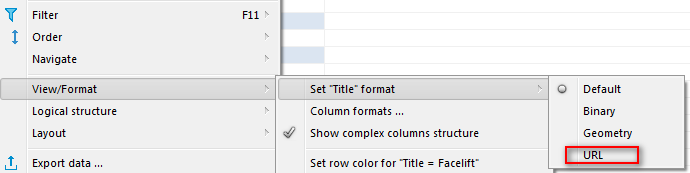
You can change the colour preferences in the Preferences window by: clicking **Window -> Preferences** on the main menu. In the window of the navigation pane on the left, expand **User Interface** and then **Appearance**, and then click **Colours and Fonts**:



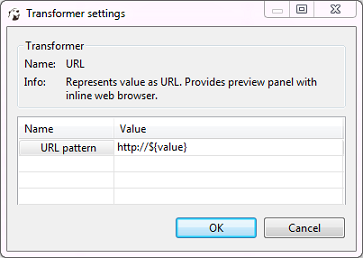
To remove the colouring by data types, on the context menu, click **View/Format -> Colourize Data Types** again.

## [Transforming Data Presentation](https://dbeaver.com/docs/wiki/Data-View-and-Format/" \l "transforming-data-presentation)

For string and numeric data types, DBeaver provides tools to transform the data presentation into a number of formats, such as URL and Binary for strings and Epoch Time, Number Radix, etc. for numbers. To change the data presentation in a certain column, right-click a cell in the column. Then, on the context menu, click **View/Format -> Set {column name} format** and click the presentation type name:



The Transformer settings window opens showing the value in the chosen format. Click **OK** to apply the change:

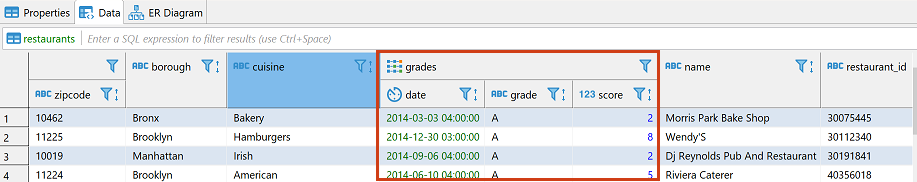


The values in the column appear in the new format.  
NOTE: For URL format, the resulting cell provides a link to the URL in a browser window.

To roll back the changes to the default format, right-click any cell in the column, and on the context menu, click **View/Format -> View as -> Default**.

## [Structurizing Complex Data Types](https://dbeaver.com/docs/wiki/Data-View-and-Format/" \l "structurizing-complex-data-types)

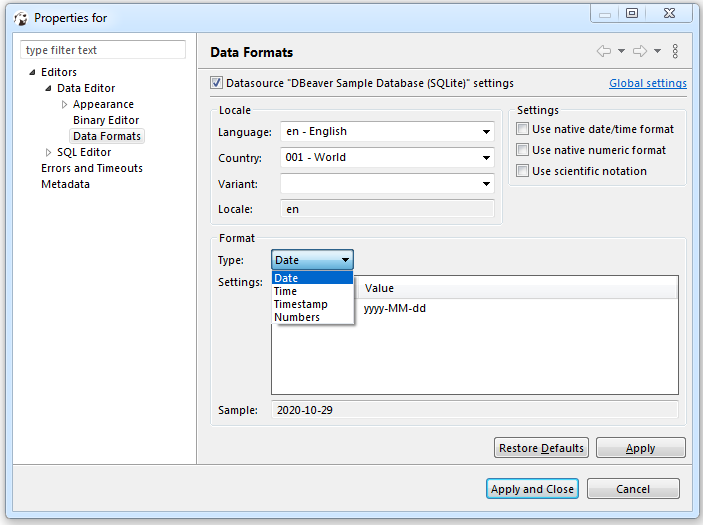
For complex data types (that themselves represent a structure), such as objects, structures and arrays, DBeaver provides a tool for breaking them into columns:



To do so, right-click a cell in the column and, on the context menu, click **View/Format -> Visualize complex columns**.

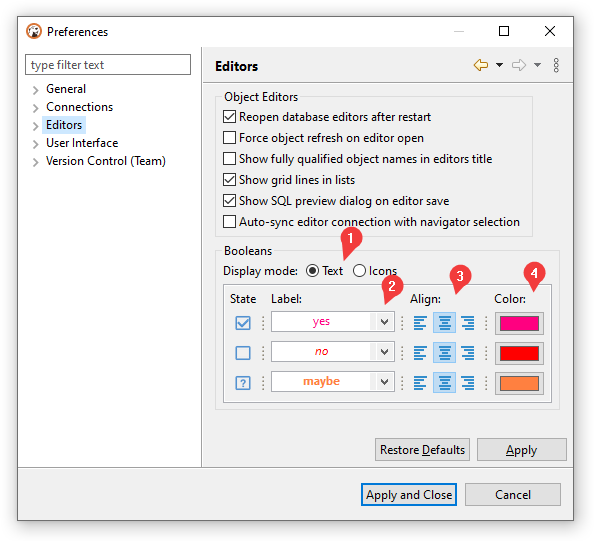
## [Configuring Numeric and Time Data Formats](https://dbeaver.com/docs/wiki/Data-View-and-Format/" \l "configuring-numeric-and-time-data-formats)

You can specify the exact format of Time, Timestamp, Date, and Number data used in the currently open database or globally. To specify a format, right-click any cell in the table and, on the context menu, click **View/Format -> Data formats**. The Properties window opens displaying the **Data Formats** page:



To configure only the format for the current database, select the **Datasource "[Connection name]" settings** checkbox. To configure the settings globally, to all databases that you have in DBeaver, click **Global settings**.  
You can specify the locale for the data format in the **Locale** area. In the **Type** dropdown list, click the name of the data type and in the **Settings** table, click the required format.  
To apply the changes and make them visible in the table, click **Apply and Close** and then refresh the window (F5).

## [Configuring Boolean presentation](https://dbeaver.com/docs/wiki/Data-View-and-Format/" \l "configuring-boolean-presentation)



You can choose between two presentation modes1:

* Text-based
* Icon-based

### [Text-based presentation](https://dbeaver.com/docs/wiki/Data-View-and-Format/" \l "text-based-presentation)

This is the most customizable mode. You can:

1. Change **labels** under Label column.  
   Otherwise, you can use presets available in Drop-down Menu2
2. Change **alignment** of value inside grid cell.3  
   Following variants are available: **left**, **center**, and **right**
3. Change **color** of value using color picker under Color column4.  
   You can reset color to match current theme's contrast color in Drop-down Menu2 ⇒ Colors ⇒ Use theme default color
4. Change font **style** in Drop-down Menu2 ⇒ Styles.  
   Following variants are available: normal, **bold**, and italic

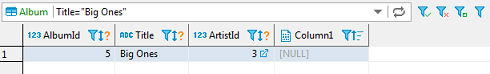
### [Icon-based presentation](https://dbeaver.com/docs/wiki/Data-View-and-Format/" \l "icon-based-presentation)

This presentation only supports alignment changing.

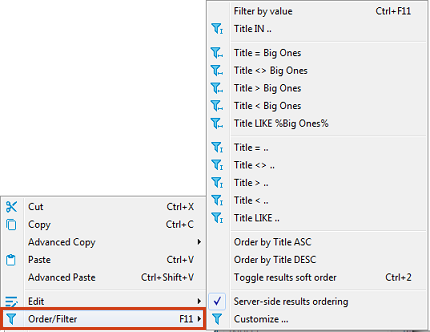
# Data Filters

You can apply custom filters to table contents or query results. There are several ways in which you can filter data in the table.

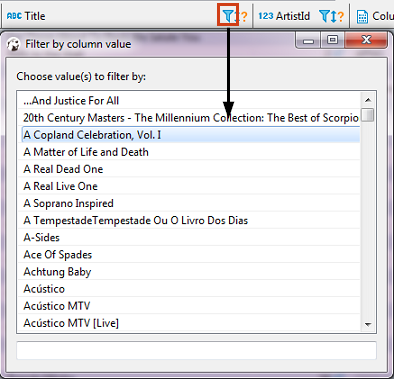
One of the ways is to use the filter field above the table next to the top toolbar. To filter data, enter an SQL expression into the field and click the Apply filter criteria button () next to the field or press Enter.



You can apply ready-to-use SQL expressions or SQL expression templates via the context menu. To select a ready SQL expression or a template, press F11 or right-click the cell, then click **Order/Filter** on the context menu and then click one of the expressions.



The third way is to filter data by a cell value using the filter icon in the column header. To filter data this way, click the filter icon in the column header and then double-click the cell value in the Filter by the column value dialog box:

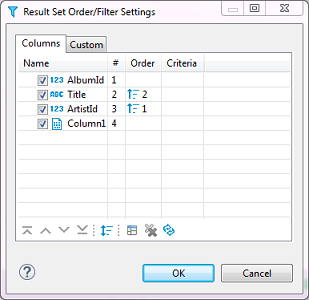


The data updates dynamically. To remove a filter, click the Remove All Filters/Orderings button () in the top toolbar of the editor.

You can save the current filter settings for the database object to apply the next time you reopen it in the editor. To save the current filter settings, click the Save filter settings for current object button () in the top toolbar.

# [Advanced filters configuration](https://dbeaver.com/docs/wiki/Data-Filters/" \l "advanced-filters-configuration)

The main tool for managing the appearance of the data table is the Result Set Order/Filter Settings window.

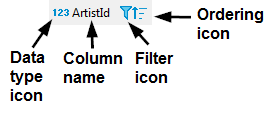


To open this window, click the Custom Filters button () in the top toolbar of the editor or click the Configure button () and then click **Order/Filter** on the dropdown menu.

The Result Set Order/Filter Settings window provides tools to:

* Order data inside columns
* Manage the display of columns in the table
* Manage the order of columns in the table
* Filter data in the table using an SQL expression

Another tool for managing the data appearance is the column headers. In the data table, every column header contains three elements which each has its own function: Data (column) type icon, column name, filter icon, and ordering icon.



* By simply clicking the column name or column type, the icon highlights the whole column.
* You can click the column type icon and then drag and drop the column to a different position in the table.
* You can click the column name and then drag the cursor to the right or left to highlight multiple columns.
* Clicking the ordering icon allows you to order the data in the column in ascending or descending order - see the 'Ordering Data in Columns' section, further in this article
* Clicking the filter icon allows you to filter the data by a cell value, see [TBA]

## [Ordering Data in Columns](https://dbeaver.com/docs/wiki/Data-Filters/" \l "ordering-data-in-columns)

You can order data in columns in one of the following ways:

1. Click the ordering icon () in the header of the column.

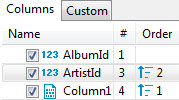


The icon has three states:

* + Clicking once establishes ascending order ()
  + Clicking a second time changes the order to descending ()
  + Clicking a third time removes the ordering from the column ()

To order data by several columns, go column by column, setting the order with the Ordering icon, starting from the column by which you want to order data first.

1. Click the Custom Filters button () in the top toolbar of the editor to open the Result Set Order/Filter Settings window (see above):

a) Next to the column by which you want to order data in the first turn, set the ascending or descending order using the same three-state principle as described above.  
b) Set the ordering in other columns by which you want to sort the data in the second, third, etc. turn. The **Order** column indicates the order in which the sorting will happen.  
NOTE: The number (**#**) column indicates the initial order of columns.  
  
c) To easily move the ordering setting from column to column, you can use the Move up/down/to top/to bottom/ buttons: 

To reset the data ordering to its initial state, click the Reset button () in the same window.

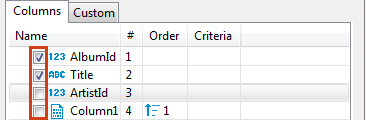
Also, to remove all ordering settings, click the Remove All Filters/Orderings button () in the top toolbar of the Data Editor.

## [Managing Display of Columns in Data Table](https://dbeaver.com/docs/wiki/Data-Filters/" \l "managing-display-of-columns-in-data-table)

To hide a single column, right-click the column or any cell in it and click **View/Format -> Hide column** on the context menu. To unhide a hidden column, open the Result Set Order/Filter Settings window (see the image at the beginning of this article) and select the checkbox next to the column name, or click the Reset button ().

To display or hide columns in the data table, in the Result Set Order/Filter Settings window:

1. Select the checkboxes next to the columns that you want to see in the table and clear the checkboxes next to those that you want to hide.



1. Use the Show All () and Show None () buttons at the bottom of the window.

## [Sorting Columns in Data Table](https://dbeaver.com/docs/wiki/Data-Filters/" \l "sorting-columns-in-data-table)

You can modify the order of columns in the data table in two ways:

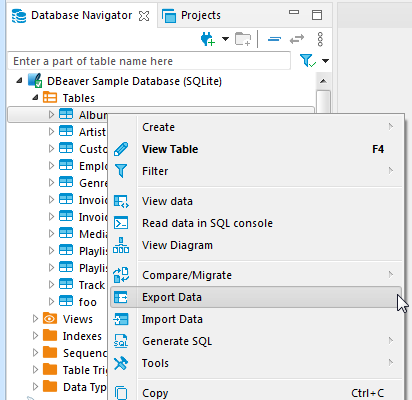
1. Click the icon in the column header and drag-and-drop the column to a new position.
2. To sort the column alphabetically, in the Result Set Order/Filter Settings window (open by clicking the Custom Filters button () in the top toolbar of the editor), click the Sort button ()
3. In the Result Set Order/Filter Settings window, click the column to set the focus to it and then move it using the navigation buttons: ()

# Data export/import

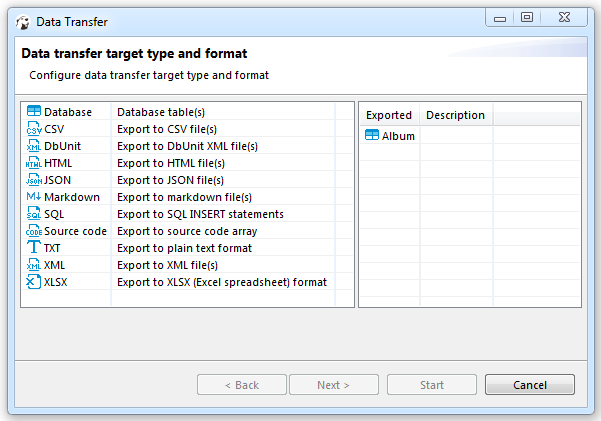
You can perform data export/import or migration for database table(s). We will describe most typically used cases.

## [Exporting table data to CSV format](https://dbeaver.com/docs/wiki/Data-transfer/" \l "exporting-table-data-to-csv-format)

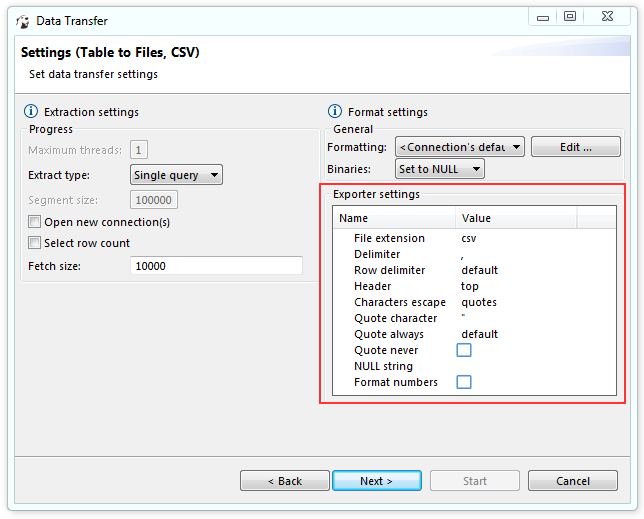
1. Select a table or tables you want to export. In the context menu choose **Export Data**.  
   (Note: you can also export data from custom SQL queries results. To do that, choose **Export data** in the results context menu).



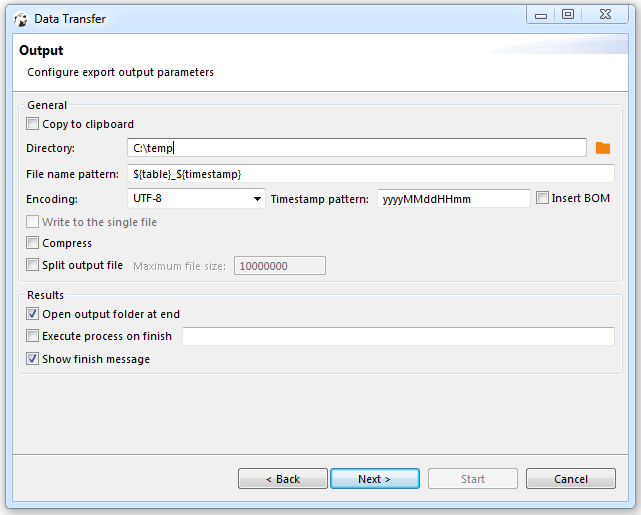
1. Choose export format. DBeaver supports many different output formats including CSV, HTML, XLSX, etc:



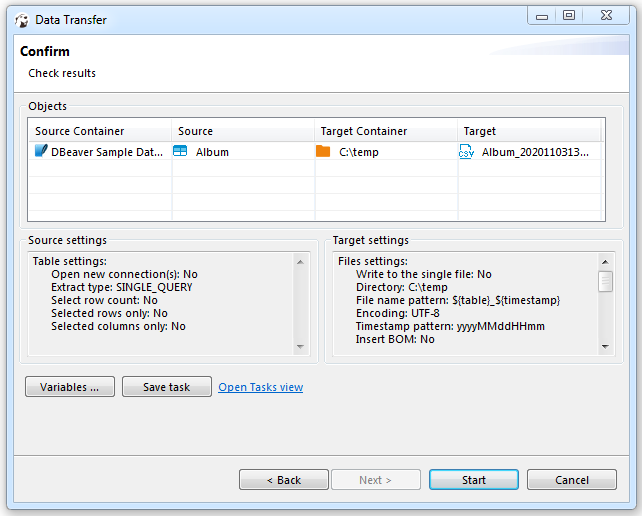
1. Set data extraction options (how the data will be read from the tables). This may affect the extraction's performance. And set export format option. They are specific to the data format you chose on step 2:



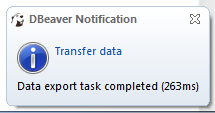
1. Set options for output files or clipboard. Note: Timestamp pattern is used here to target the file name pattern:



1. Review what you want to format and into which format you will export it. You can also save all your settings as a task in this step or change the task variables:



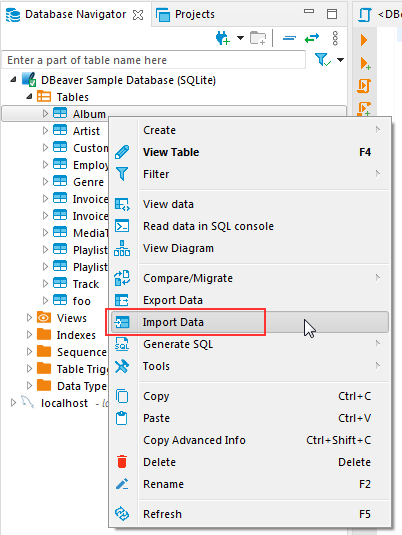
1. Press finish. See extraction progress. You can keep working with your database during the export process as the extraction will be performed in the background. Note: avoid changing data in tables you have selected to be exported while the exporting is in progress. In the end you will see status message:



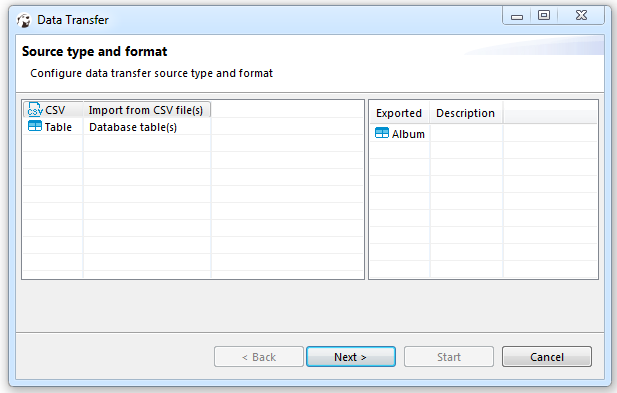
## [Importing data from CSV format](https://dbeaver.com/docs/wiki/Data-transfer/" \l "importing-data-from-csv-format)

You can import data from CSV file(s) directly into your database table(s).

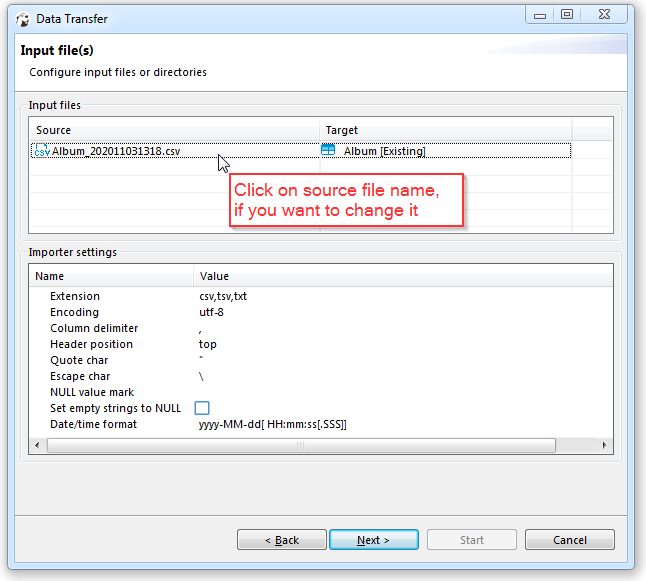
1. Select a table(s) to which you want to import data. In the context menu choose **Import Data**:



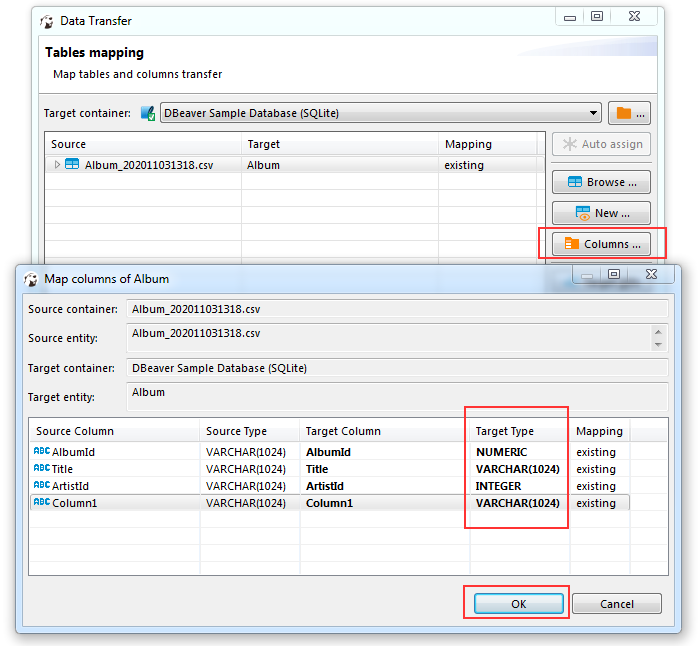
1. Choose import format (CSV):



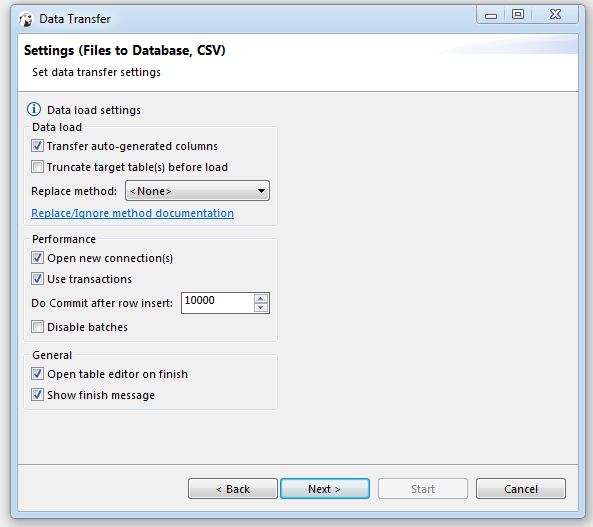
1. Select input CSV file for each table you want to import and you can change the Importer settings (format specific) at this step:



1. Set CSV-to-table mappings. You need to set a column in the CSV file for each database table column. You can skip columns (the value will be set to NULL in the target table column). You can set constant values for the table column if there is no source column for it in the CSV.

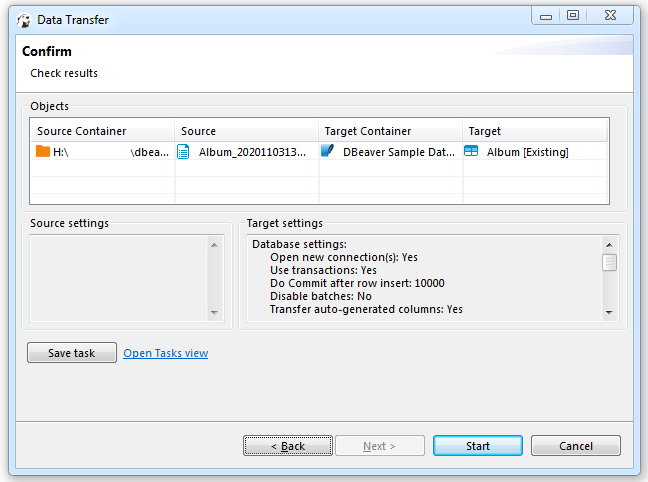


1. Set options for loading data in the database. These options may affect the loading's performance:

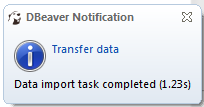


About the replacing method option, you can [read here](https://dbeaver.com/docs/wiki/Data-transfer/Data-Import-and-Replace).

1. Review which file(s) and to which table(s) you will import. You can also save all your settings as a task in this step:

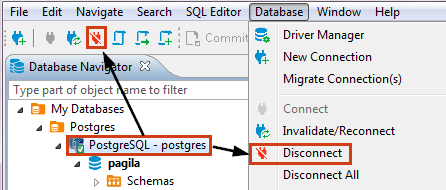


1. Press finish. See extraction progress. You can keep working with your database during the export process as the data loading will be performed in the background. Note: avoid changing data in tables you have selected to be imported while the import is in progress. In the end you will see the status message:



# Disconnect from Database

You might need to disconnect from a database to free up resources or close transactions. To disconnect from a database, click the connection in the [Database Navigator](https://dbeaver.com/docs/wiki/Disconnect-from-Database/Database-Navigator) or [Projects](https://dbeaver.com/docs/wiki/Disconnect-from-Database/Projects) view, and then click the Disconnect button in the toolbar or click **Database -> Disconnect** on the main menu:



You can also right-click the connection and click **Disconnect** on the context menu.

NOTE: The Disconnect button and menu items are available only for those connections that are activated, that is, marked with the connected sign: .

When DBeaver disconnects from a database, its icon changes to its original state (not connected), for example,  for PostgreSQL database.

To disconnect from all active connections, click **Database -> Disconnect All** on the main menu.

# Data Refresh

In the bottom left part of the Data editor, you can find the toolbar with actions: image

Refreshing is necessary if the database contains changes made by other users working on it simultaneously with you, and you want to see them in your DBeaver window. To refresh data manually, right-click anywhere in the data table and click image on the context menu or press F5.

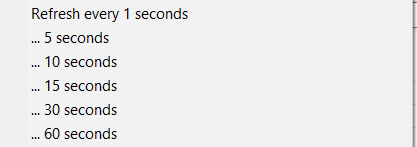
## [Autorefresh](https://dbeaver.com/docs/wiki/Data-Refresh/" \l "autorefresh)

You can also schedule auto-refresh to happen regularly. To auto-refresh the database on schedule:

Click on the arrow in the auto-refresh button image. Now you can configure auto refresh in the two following ways:

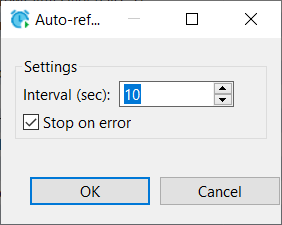
### [1. Use a predetermined duration](https://dbeaver.com/docs/wiki/Data-Refresh/" \l "1.-use-a-predetermined-duration)

You can select via choosing an available predetermined period in the context menu.



### [2. Set your duration](https://dbeaver.com/docs/wiki/Data-Refresh/" \l "2.-set-your-duration)

You can set your time duration between refreshes. To do that, select image.The Auto-refresh configuration dialog box will open:



a) Set **Interval** in seconds.  
b) Select the **Stop on error** checkbox if you want the refresh to stop when it encounters an error or clear it if the refresh should ignore mistakes.  
c) Click **OK**.

When you perform either of these two alternative options above, the system starts refreshing the data as scheduled, and the Configure auto-refresh button changes to the **Stop auto-refresh** button image. To stop the auto-refresh, click the **Stop auto-refresh** button or the arrow next to it and click **Stop** on the auto-refresh menu.

**Feedback:**

First of all, I am A Noorjahan Begum, I am very much thankful to Prof. Ch.

Radhakumari madam for the

wonderful opportunity she gave to upgrade my skills. SURE Trust is doing a great service by providing free courses to students. During the course we not

only learned the skills but also discipline, dedication and attentiveness as well.

SURE Trust is also conducting LST Sessions on Sundays which is more useful for us in our day today life.

Our Trainer Mr. Ashwani Ahlawat is one of the best we got as he teaches the

course in a very understandable manner. Mr. Ashwani Ahlawat is always on time for the session and he never skipped any classes. Mr. Ashwani Ahlawat probes us with questions during the session and clears our doubts when we are unable to understand on the topic. Mr. Ashwani Ahlawat gave us assignments on every topic and helps us to finish it if we are stuck or unable to

complete it. Thank you so much Mr. Ashwani Ahlawat

Heartly wishing SURE TRUST & Mr. Ashwani Ahlawat for their future endeavors

* 1. Uniqueness of the Course
* Courses are thought practically and explained in detail.

\*Machine Learning is the goal of our course and it was thought in an easy understandable manner.

* Assignments are given on every topic which made us to learn.
* Concentrated on every student and thought us discipline along with the course.
* Made us to prepare the student’s report which will be helpful for our future reference.
  1. Concluding Remarks

I am very much happy that I have learnt the course successfully. I am blessed

to learn the course in SURE TRUST. I would like to thank each and every person

of SURE TRUST for providing me an opportunity to learn the course. SURE TRUST is helping the students to gain the knowledge on latest technology and

helps them to get good jobs. I am proud to be a student of SURE TRUST.