

Data



The [Data](#) module, located in the [Reporting group](#), provides the ability to retrieve and export data from the Fishbowl database. All data can be viewed, but the database cannot be modified. The admin user will always have access to the [Data](#) module. Other users can be given access in the [User Group module](#). A query can be [run](#), [saved](#), [exported](#), and [scheduled](#). The query results can also be accessed through the [Fishbowl API](#).

Contents

- [Run a query](#)
- [Query parameters](#)
- [Save a query](#)
- [Filter a query](#)
- [Export a query](#)
- [Copy data](#)
- [Query history](#)
- [Schedule a query](#)
- [Extract a query](#)
- [Database tables](#)
- [Server log](#)
- [MODULE OPTIONS](#)

[Back to all categories](#)

Reporting

[Training Videos](#)

[BI Report](#)

[BI Editor](#)

[Data](#)

[Reports](#)

[Advanced Reports](#)

[Forecast](#)

Run a query

1. Open the [Data](#) module and click the **New** button located on the module toolbar.

2. Type any [SQL](#) query, for example, **SELECT * FROM Customer**

The screenshot shows a software interface with three tabs: 'General', 'Details', and 'Tables'. The 'General' tab is active, displaying a 'SQL' section with a text area containing the query '1 SELECT * FROM customer'. Below the query, there is a 'Data' section showing a table with two columns: 'NAME' and 'NUMBER'. The table contains four rows of data: 'Peak Mountaineer' (165), 'Shawn Sheffer' (166), 'Faster Bicycles' (167), and 'Disney's Bikes' (168). The 'Shawn Sheffer' row is highlighted. To the right of the table are icons for 'RLS' and 'CSV'. At the bottom of the 'Data' section, there is a checkbox for 'Raw data' and a 'Row Count: 72' indicator.

NAME	NUMBER
Peak Mountaineer	165
Shawn Sheffer	166
Faster Bicycles	167
Disney's Bikes	168

3. To see the results, click the **Run Query** button on the toolbar to the right, or press **Ctrl + Enter**

The columns in the **Data** section can be [dragged, sorted, and hidden](#).

Query parameters

The **Data** module supports several query parameters. Parameters make it easier to select data from the database. For example, instead of manually entering the ID of a customer directly into a query, a parameter allows the customer to be selected from a drop down list.

Query without parameters

```
1 select * from customer where id = 173
```

Query with parameters

The screenshot shows a query editor with the query '1 select * from customer where id = \$CUST[Select_a_customer]'. A 'Query Parameters' dialog box is open, showing a dropdown menu with 'Allan's Raceway' selected. The dialog box has 'OK' and 'Cancel' buttons.

Below is a list of the available parameters.

- \$TEXT{text}
- \$NUM{number}
- \$QTY{quantity}
- \$DATE{date}

- \$RANGE{date_range}
- \$PART{Part}
- \$PROD{Product}
- \$LOC{Location}
- \$LG{Location_Group}
- \$BOM{Bill_of_Materials}
- \$CAR{Carrier}
- \$CUST{Customer}
- \$CG{Customer_Group}
- \$SHIP{Shipment}
- \$VEND{Vendor}
- \$SO{Sales_Order}
- \$PO{Purchase_Order}
- \$TO{Transfer_Order}
- \$MO{Manufacture_Order}
- \$WO{Work_Order}

Below are some examples of queries that use parameters.

```
select * from part where
datelastmodified between
$RANGE{Modified_Date}
select * from location where
```

locationgroupid=\$LG{Location_Group}

Notes

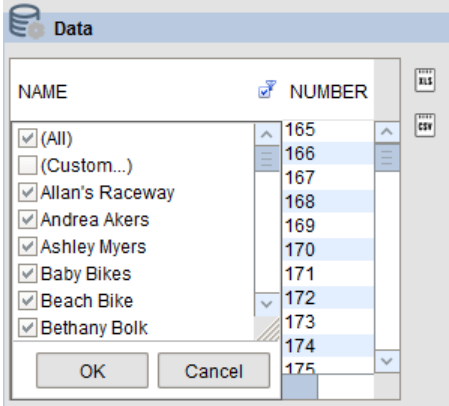
- All of the lookup parameters (the lower section of the parameter list) return the database ID. For example, **\$CUST** will insert the **ID** of the selected customer into the query, not the **NAME** of the customer.
- The text between the curly braces { } is the label that will be displayed next to the field.
- Make sure to use curly braces { } around the label and not parentheses ().
- The text in the label can be modified to provide a more informative description. Use an underscore _ to display a space in the label.
- Because parameters don't have default values, any parameter used in the query will require a value to be entered (except for **\$TEXT**).
- Because parameters require user input, queries with parameters cannot be used in the [Schedule module](#).

Save a query

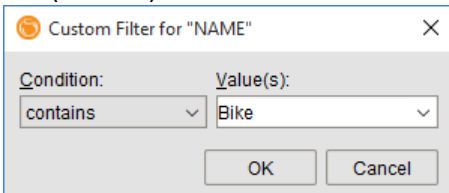
1. To save a query for reuse, enter a name in the **Name** field. The description is optional.
2. Click the **Save** button on the module toolbar, or press **Ctrl + S**

Filter a query

- The data returned from a query can be further filtered by clicking on the right side of any column header.



- Click **(All)** to select or unselect all entries.
- Click an individual entry to show or hide it in the results.
- Click **(Custom...)** to add a custom filter to the results.



Possible filters include **is anything**, **is**, **doesn't equal**, **is in**, **isn't in**, **is empty**, **is not empty**, **begins with**, **ends with**, **contains**, and **doesn't contain**.

- When data is exported, only the filtered data that is currently visible in the table will be exported.
- Filters are temporary. The next time the query is executed, the filters will be reset.

Export a query

1. After [running the query](#) and applying any [filters](#), click the **Export to .xls** button or the **Export to .csv** button.
2. Choose where the file should be saved and then click **Export**.
3. Browse to the file and open it with a spreadsheet application or a text editor.

Copy data

The text in the **Data** section can also be copied and pasted into a text document or a spreadsheet.

NAME	NUMBER
Peak Mountaineer	165
Shawn Sheffer	166
Faster Bicycles	167
Disney's Bikes	168
Andrea Akers	169
Kathy Thoughtful	170
Linkin Bicycles	171
William Blake	172

☐ Raw data Row Count: 72

- To copy a single cell, double-click the cell and then press **Ctrl + C**.
- To copy a single row, click the row and then press **Ctrl + C**.
- To copy all rows, press **Ctrl + A** and then press **Ctrl + C**.
- To copy consecutive rows, click and drag the mouse over the rows and then press **Ctrl + C**.
- To copy non-consecutive rows, hold down the **Ctrl** key while clicking each row and then press **Ctrl + C**.

Query history

- To move through the history of executed queries, click the **Previous** button or the **Next** button on the toolbar to the right.
- The history is deleted each time the module is closed.
- The number of items stored in the history can be set in the [Data module options](#).
- The history only displays queries that have been executed. To undo a typing mistake, press **Ctrl + Z**.

Schedule a query

1. Open the [Schedule module](#) and click the **New** button.
2. Select **Data** and then click **Next**.

Schedule Type

Select the type of scheduled event you would like to create.

Database Backup

Backup the Fishbowl database

Export

Export selected data to CSV

Import

Import data from CSV

Java Class

Run a customized Java class

Report

Run a selected report

Data




Run a data export query


3. Select a saved query, enter the required information, and then click **Next**.

4. Set the scheduled time and finish the wizard.

Extract a query

The Fishbowl Server can log many of the queries that are run when a user is using Fishbowl. It is often helpful to start with a query from a report and then modify it as necessary. Below are the steps for extracting a query for inventory quantities.

1. In a Fishbowl Client, run the **Inventory Availability** report.
2. Open the  **Data** module and click the  **Server Log** button.
3. Click the **Reports** tab.
4. Scroll to the bottom of the tab and copy the query starting right after **SQL query string:** until the end of the file.
5. Paste the query into the  **Data** module.
6. Use the parameters listed below the query to replace the ? symbol.

Below is an example of how to modify the query. After completing the modification, press the  **Run Query** button.

```

SELECT FIRST 1000 SKIP 0 part.num AS part, part.description, uom.code AS uomcode,
company.name AS company,
  COALESCE((SELECT SUM(qtyonhand.qty)
            FROM qtyonhand
            WHERE qtyonhand.partid = part.id
              AND qtyonhand.locationgroupid IN (1,2,3,4)), 0) AS qty,
  COALESCE((SELECT SUM(qtynotavailable.qty)
            FROM qtynotavailable
            WHERE qtynotavailable.partid = part.id
              AND qtynotavailable.locationgroupid IN (1,2,3,4)), 0) AS
Unavailable,
  COALESCE((SELECT SUM(qtydropship.qty)
            FROM qtydropship
            WHERE qtydropship.partid = part.id
              AND qtydropship.locationgroupid IN (1,2,3,4)), 0) AS DropShip,
  COALESCE((SELECT SUM(qtycommitted.qty)
            FROM qtycommitted
            WHERE qtycommitted.partid = part.id
              AND qtycommitted.locationgroupid IN (1,2,3,4)), 0) AS
qtycommitted,
  COALESCE((SELECT SUM(QtyAllocated.qty)
            FROM QtyAllocated
            WHERE QtyAllocated.partID = Part.id
              AND QtyAllocated.locationgroupid IN (1,2,3,4)), 0) AS allocated,
  COALESCE((SELECT SUM(qtyonorder.qty)
            FROM qtyonorder
            WHERE qtyonorder.partid = part.id
              AND qtyonorder.locationgroupid IN (1,2,3,4)), 0) AS onorder

FROM part
  LEFT JOIN uom ON part.uomid = uom.id
  JOIN company ON company.id = 1

```

```

WHERE part.num LIKE ?
AND part.id != 0
AND part.typeid = 10
AND part.activeflag = 1

```

```
ORDER BY part.num
```

```

06 Nov 2019 11:52:27 DEBUG [InventoryAvailability subreports #1]
jasperreports.engine.query.JRJdbcQueryExecuter (JRJdbcQueryExecuter.java:396)
- Parameter #1 (partNum of type java.lang.String): %

```

```

SELECT FIRST 1000 SKIP 0 part.num AS part, part.description, uom.code AS uomcode,
company.name AS company,
  COALESCE((SELECT SUM(qtyonhand.qty)
            FROM qtyonhand
            WHERE qtyonhand.partid = part.id
              AND qtyonhand.locationgroupid IN (1,2,3,4)), 0) AS qty,
  COALESCE((SELECT SUM(qtynotavailable.qty)
            FROM qtynotavailable
            WHERE qtynotavailable.partid = part.id
              AND qtynotavailable.locationgroupid IN (1,2,3,4)), 0) AS
Unavailable,
  COALESCE((SELECT SUM(qtydropship.qty)
            FROM qtydropship
            WHERE qtydropship.partid = part.id
              AND qtydropship.locationgroupid IN (1,2,3,4)), 0) AS DropShip,
  COALESCE((SELECT SUM(qtycommitted.qty)
            FROM qtycommitted
            WHERE qtycommitted.partid = part.id
              AND qtycommitted.locationgroupid IN (1,2,3,4)), 0) AS
qtycommitted,
  COALESCE((SELECT SUM(QtyAllocated.qty)

```

```

COALESCE((SELECT SUM(qtyallocated.qty)
          FROM QtyAllocated
          WHERE QtyAllocated.partID = Part.id
                AND QtyAllocated.locationgroupid IN (1,2,3,4)), 0) AS allocated,
COALESCE((SELECT SUM(qtyonorder.qty)
          FROM qtyonorder
          WHERE qtyonorder.partid = part.id
                AND qtyonorder.locationgroupid IN (1,2,3,4)), 0) AS onorder

FROM part
LEFT JOIN uom ON part.uomid = uom.id
JOIN company ON company.id = 1

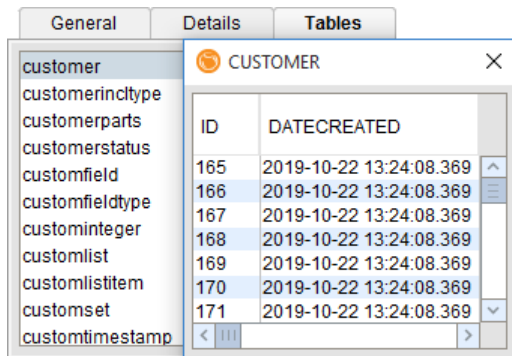
WHERE part.num LIKE '%'
AND part.id != 0
AND part.typeid = 10
AND part.activeflag = 1

ORDER BY part.num

```

Database tables

Click the **Tables** tab to see a list of all of the tables, fields, and data in the Fishbowl database. Double-click a table name to see it in a separate window.



A database dictionary can also be downloaded by clicking [here](#).