Learn how to use OLIB report sections.

## Create a Section

With the report in modify mode, click on **New** under the **Report Sections** list attribute to enter the details for a section. The following page will be offered:

### Report Section

<table>
<thead>
<tr>
<th>Report Description</th>
<th>CM004 - Items Never Borrowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section Seq.</td>
<td>1</td>
</tr>
<tr>
<td>Section No.</td>
<td>2</td>
</tr>
<tr>
<td>Identifier</td>
<td>Page</td>
</tr>
<tr>
<td>Width</td>
<td>21.0cm</td>
</tr>
<tr>
<td>Height</td>
<td>29.7cm</td>
</tr>
<tr>
<td>A4 Portrait</td>
<td>Portrait&lt;&gt; Landscape</td>
</tr>
</tbody>
</table>

### Margins

- **Top**: 7.7 cm
- **Left**: 7.7 cm
- **Right**: 7.7 cm
- **Bottom**: 7.7 cm

### Header

- **Text**: #SYSDATE.DD-Mon-YYYY HH24.MI#(tab)#PAGE_NR#/#PAGE_COUNT#
- **Header Style**: Report Page Header
- **Line Below?**: No

### Content

- **Query**

### Result Columns

<table>
<thead>
<tr>
<th>Label</th>
<th>Type</th>
<th>Format</th>
<th>Heading</th>
<th>Style</th>
<th>%</th>
<th>Group</th>
<th>Data Style</th>
<th>Line</th>
</tr>
</thead>
</table>

### Footer

- **Text**: (tab)Copyright (c) OCLC 2017
- **Footer Style**: Report Page Header
- **Line Above Footer?**: No

### Report Section

This portion of the page identifies the section itself. Firstly, the report’s description will be shown, for informational
purposes.

Below this are:

Section Seq.: this is the sequence of this section for this report. It can be manually edited. The first section for a report must have a ‘1’ here and there must be no gaps in the sequencing, otherwise the adjustment cannot be reliably made on the Report Definition sheet using the Up and Down process.

Section No.: this is a unique identifier in the database for this Report Section record. It should not be editable

Identifier: this is a user entered text label for the section to aid linking (in future developments) and identification of this section when viewed on the report definition screen.

Width and Height: these fields determine the width and height of this section respectively. It is a free text entry field for a number which will default to be understood as centimetres to specify the width and height of the section.

The values for the first section are used to determine the page size for the whole report. The helper buttons provide for a simple way to specify A4 and Portrait/Landscape mode. Optionally, the A4 button can be swapped for a Letter size button helper by amending the layout. There is only one Report Section layout.

To specify a size in another unit, the number can be followed by "pt" for points; "mm" for millimetres; "in" for inches or "px" for pixels.

When a value is entered, the sizing in centimetres will be shown after the input field. If not specified, OLIB Reports will presume that the report is A4.

To clear a value from the width, height or margin field, enter a “-”.

Margins

These four fields specify the section margins. For the first section, this will determine the page margins. The input is handled in the same way as the width and height entry fields described above.

The default margins for A4 are 0.64cm for left and right and 1.27cm for top and bottom.

Section margins are not currently implemented. The intention is to provide support for label printing and, as such, section margins may prove essential for that process.

Example First Section – Page Setup

As mentioned above, the first section will be used to determine the page size, header and footer. This can be as simple as entering a Section Seq., and an Identifier and then clicking A4 Portrait. For example:
To produce a more formal report, a header and footer can be added to this section and will be treated as the page header and footer during PDF production.

**Header**

The Text, Header Style and Line Below? fields provide for text to appear at the top of the section. For the first section, this will be treated as the page header.

There are several pieces of automatic text that can be used here:

- **#SYSDATE:format#**
  - This will be used as a placeholder to present the current date in the Oracle format given

- **{tab}**
  - This will be used to separate the page header text into 3 portions. The first will be left justified at the left of the page, the second will be centrally aligned in the middle of the page, and the third will be right justified on the right side of the page.
  - If no “{tab}” is used, the header will appear on the left
  - The portions are all optional. Therefore, specifying “{tab}” at the start will place the following portion in the centre, with no text on the left

- **#PAGE_NR#**
  - This will be replaced with the current page number

- **#PAGE_COUNT#**
  - This will be replaced with the number of pages

- **{nl}**
  - This will be replaced with a new line. This is likely more useful in heading text than for a page header.
• #n#
  ◦ This can be used as a placeholder for the \( n \)th parameter value
• $n$
  ◦ Where possible, this can be used as a placeholder for the output of the \( n \)th column in the query

Either type the auto-text directly into the Text field, or select from the drop-down list to the right of the Text field. This will paste the selected auto-text at the end of whatever is currently in the Text field. If this is not the correct place for it, use the cut/paste options to move it to the required position. Note, however, that the auto-text will only be pasted into the Text field if you change the value in the drop-down list. Thus, for example, if you want to enter 2 \{tab\}'s in a row, after selecting the “tab” option for the first time, select the blank option at the bottom of the drop-down list before selecting the “tab” option for the second time.

The Style field is used to determine the font, size and colour, together with bold or italic decoration. The colour is taken from the normal mode colour settings from the OLIB Style record. Further information on the updates to the style information can be found later in this document – see Appendix A - Updates to Style Records on page 32.

The footer is defined in the same way lower down on screen.

**Content**

This section defines the body of the section, between the Header and Footer.

**Query**

• This field holds the query for the section which provides for the content of the report itself.
• Parameter handling is incorporated using the \#n\# notation, as described above.
• The query results themselves are optionally divided into four sections:
  ◦ Start of a group
  ◦ Content
  ◦ End of the group – corresponding to the appropriate group start
  ◦ Other required columns
• Further information on designing your query is given below.

**Result Columns**

This attribute lists the column definitions for the query. The **Generate Columns** button will examine the query and populate the column definitions below. These columns cannot be altered as they are determined by the query that has been written. If changes are made to the query that affect the number and/or type of columns, then the Generate Columns button must be used again.

By default, the generation of the columns records will recognise three key categories of column:

• Number
• Date
• Text
The first two values for each column record – Label and Type – are populated automatically and cannot be modified.

Format:

- This field is used for number and date columns. It specifies the format that will be applied to the value retrieved.
- If a change to the query results in a change to this column’s category, thereby rendering the format invalid, it will be nulled.
- For a date column, the session default format will be placed into this field as a starting point.

Heading:

- This optional field provides for text to appear at the top of the column or, for grouped results, preceding the value returned.

Style:

- This determines the style to be used for the heading text.

%:

- This is defaulted to evenly spread the columns to 100% of the section width, but can be amended as required.
- For a grouped query, this could be 100% for the grouping columns. See the example below for further explanation.
- A zero here will prevent the data from appearing in the output. This is useful for hiding columns that have only been included for sorting purposes when using DISTINCT.

Group:

- This determines whether this column is to represent a grouped output. For a grouped output, the value is presented only when it is changed or, for end groups, when the corresponding start group value changes.
- If there are any end group columns required (for example for totals or averages), there must be the same number of end groups as start groups, and they must be presented in reverse order. This should become clearer when reviewing the example below.

Data Style:

- This determines the style to be used for the data retrieved from the database for this column. It is not used for grouped columns as the value utilises the same style as the heading.

Line:

- For grouped columns, whether to draw a line under the start group value or over the end group value.

Line Between Records?

This option is for PDF production only. By default, the PDF output will apply alternate shading to the rows retrieved within each group. Where the output for an individual row spans more than line in the resulting report, only the first line or alternate records attract the shading. For such reports, it may be preferable to include a line between each record instead of the alternate shading. Specifying a Yes here will achieve this.
Alternative shading will be applied to all lines in the record when HTML output is chosen and, as such, this setting is ignored.

**Example Second Section - Parameter Display**

It is often useful to the recipient of the report, and for archiving purposes, to know what the values of the parameters were for the given report. This can be achieved with a query such as below:

```
Header

Text: Sample Report
Header Style: Report Header

Content

Query:
SELECT to_date(#1#, 'DD-MM-YYYY') fromdate,
to_date(#2#, 'DD-MM-YYYY') todate,
CASE WHEN #3# is null THEN 'All locations' ELSE
(SELECT LISTAGG(l.locd, ', ') WITHIN GROUP (ORDER BY UPPER(l.locd)) LOC
FROM locs l
WHERE loc in csv(#3#))
END loc
FROM dual
```

(Note that each of the Styles specified are Report Parameters)

As there must be at least one ungrouped column displayed, the Locations column makes sense in this context as it will likely have to longest value.

**Example Grouped Query Section**

The following query presents a report on join dates for users by location and category and provides a count of the users who joined each location within the period specified by the parameters. The report that is being described here will produce output such as:

```
<table>
<thead>
<tr>
<th>Label</th>
<th>Type</th>
<th>Format</th>
<th>Heading</th>
<th>Style</th>
<th>%</th>
<th>Group</th>
<th>Data Style</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>FROMDATE</td>
<td>Date</td>
<td>DD-Mon-YYYY</td>
<td>Join date from:</td>
<td>Report F</td>
<td>10</td>
<td>Ye</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TODATE</td>
<td>Date</td>
<td>DD-Mon-YYYY</td>
<td>Join date to:</td>
<td>Report F</td>
<td>10</td>
<td>Ye</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOCS</td>
<td>Text</td>
<td></td>
<td>Locations:</td>
<td>Report F</td>
<td>10</td>
<td>Ni</td>
<td>Report F</td>
<td></td>
</tr>
</tbody>
</table>
```

https://help.oclc.org/Library_Management/OLIB/Reports/Use_OLIB_reports/040OLIB_report_sections

### Sample Report

**Join date from:** 16-Sep-2010  
**Join date to:** 04-Jul-2017  
**Locations:**  
All locations  
**Location:** Branch 1

<table>
<thead>
<tr>
<th>Month Joined</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>September, 2013</td>
<td>1</td>
</tr>
<tr>
<td>February, 2017</td>
<td>1</td>
</tr>
</tbody>
</table>

**Total for category Manager:** 2

**Location:** System Location, 16th September, 2010 - 4th July, 2017 at Branch 1: 2

<table>
<thead>
<tr>
<th>Month Joined</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>September, 2013</td>
<td>1</td>
</tr>
<tr>
<td>October, 2016</td>
<td>1</td>
</tr>
</tbody>
</table>

**Total for category Manager:** 2

---

https://help.oclc.org/Library_Management/OLIB/Reports/Use_OLIB_reports/040OLIB_report_sections
The page header, footer and the parameter output has been described above. The complex query that is constructed below comprises of several components:

1. Group start columns (1,2)
   These are for the Locations and User Categories
2. Content (3,4)
   This shows the actual report data (Month joined and count)
3. Group end columns (5,6)
   These are for the User Categories and Locations, respectively
4. Sorting column (7)
   This is required when DISTINCT is utilised
5. Parameter display column (8)
   This is used to generate a friendly value for the date parameter input for usage in the End Group lines.
6. Table joins
   To connect the records in the tables involved in the query
7. Data selection
   This incorporates the parameter values to limit the output
8. Sorting
   This is required firstly to facilitate the grouping operations and then for logical flow for the viewer

### Group Start Columns

The query starts with the Location description and User Category to present a group header for the content to be produced. These columns are treated as Group Start columns as they are presented prior to the content columns.

```sql
SELECT l.locld, c.borcatld, ...
```

The column definitions for these specify:

<table>
<thead>
<tr>
<th>HEADING</th>
<th>STYLE</th>
<th>%</th>
<th>GROUP</th>
<th>DATA STYLE</th>
<th>LINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location:</td>
<td>Report First Grouping</td>
<td>100</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>User Category:</td>
<td>Report Second</td>
<td>100</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>
Content columns

The data to appear in the table is retrieved next:

```sql
SELECT l.locld, c.borcatld,
b.joindate,
COUNT(*) OVER
(PARTITION BY b.loc, b.borcat,
to_char(b.joindate, 'YYYY-MM')) cnt,
...
```

---

End Group Columns

If there are any end group columns, there must be one end group column for each start group column and these must appear in reverse order from the start group columns:

```sql
SELECT l.locld, c.borcatld,
b.joindate,
COUNT(*) OVER
(PARTITION BY b.loc, b.borcat,
to_char(b.joindate, 'YYYY-MM')) cnt,
```
COUNT(*) OVER (PARTITION BY b.loc, b.borcat) cnt4cat,
COUNT(*) OVER (PARTITION BY b.loc) cnt4loc

...  

<table>
<thead>
<tr>
<th>HEADING</th>
<th>STYLE</th>
<th>%</th>
<th>GROUP</th>
<th>DATA STYLE</th>
<th>LINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total for category $2$:</td>
<td>Report End Second Grouping</td>
<td>100</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Total for $8$ at $1$:</td>
<td>Report End First Grouping</td>
<td>100</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

*Column 8 is defined below.*

**Join clauses**

The query retrieves data from three tables and will need to link the records together as below:

```sql
SELECT l.locld, c.borcatld,
       b.joindate,
       COUNT(*) OVER
           (PARTITION BY b.loc, b.borcat,
            to_char(b.joindate, 'YYYY-MM')) cnt,
       COUNT(*) OVER (PARTITION BY b.loc, b.borcat) cnt4cat,
       COUNT(*) OVER (PARTITION BY b.loc) cnt4loc
FROM borrowers b, locs l, borcats c
WHERE b.loc = l.loc
     AND b.borcat = c.borcat
```

...  

**Data Selection by Parameter Value**

To be able to limit the query by the parameter values entered, additional clauses are required:

```sql
SELECT l.locld, c.borcatld,
       b.joindate,
       COUNT(*) OVER
           (PARTITION BY b.loc, b.borcat,
            to_char(b.joindate, 'YYYY-MM')) cnt,
       COUNT(*) OVER (PARTITION BY b.loc, b.borcat) cnt4cat,
       COUNT(*) OVER (PARTITION BY b.loc) cnt4loc
FROM borrowers b, locs l, borcats c
WHERE b.loc = l.loc
     AND b.borcat = c.borcat
```
(PARTITION BY b.loc, b.borcat,
to_char(b.joindate, 'YYYY-MM')) cnt,
COUNT(*) OVER (PARTITION BY b.loc, b.borcat) cnt4cat,
COUNT(*) OVER (PARTITION BY b.loc) cnt4loc
FROM borrowers b, locs l, borcats c
WHERE b.loc = l.loc
    AND b.borcat = c.borcat
    AND b.joindate BETWEEN to_date(#1#, 'DD-MM-YYYY')
    AND to_date(#2#, 'DD-MM-YYYY')
    AND (#3# IS NULL OR b.loc IN csv[#3#])

... Make a note of the Date format specified here (in red).

Sorting the results

For the grouping to work correctly, the results must be sorted by the grouping start columns. Additionally, for legibility, the query can be sorted within the group to present the data in a sensible order:

SELECT l.locld, c.borcatld,
    b.joindate,
    COUNT(*) OVER
        (PARTITION BY b.loc, b.borcat,
        to_char(b.joindate, 'YYYY-MM')) cnt,
    COUNT(*) OVER (PARTITION BY b.loc, b.borcat) cnt4cat,
    COUNT(*) OVER (PARTITION BY b.loc) cnt4loc
FROM borrowers b, locs l, borcats c
WHERE b.loc = l.loc
    AND b.borcat = c.borcat
    AND b.joindate BETWEEN to_date(#1#, 'DD-MM-YYYY')
    AND to_date(#2#, 'DD-MM-YYYY')
    AND (#3# IS NULL OR b.loc IN csv[#3#])
ORDER BY UPPER(l.locld), UPPER(c.borcatld), b.joindate

Note that no semicolon should be placed at the end of the statement.

Final touches

One complication with the above query is that it will retrieve one record for every matching user. To avoid that the keyword DISTINCT (or UNIQUE) must be placed at the start of the query, after SELECT. This introduces a complication.
in that the value for joindate will provide for granularity to the second the user joined, rather than the simpler month and year being reported and intended for grouping. A suitable replacement should be used, with a duplication of this column for sorting purposes:

```
SELECT DISTINCT l.locld, c.borcatld,
   to_char(b.joindate, 'FMMonth, YYYY') showdate,
   COUNT(*) OVER (PARTITION BY b.loc, b.borcat,
   to_char(b.joindate, 'YYYY-MM')) cnt,
   COUNT(*) OVER (PARTITION BY b.loc, b.borcat) cnt4cat,
   COUNT(*) OVER (PARTITION BY b.loc) cnt4loc,
   to_char(b.joindate, 'YYYY-MM') sortdate
FROM borrowers b, locs l, borcats c
WHERE b.loc = l.loc
AND b.borcat = c.borcat
AND b.joindate BETWEEN to_date(#1#, 'DD-MM-YYYY')
AND to_date(#2#, 'DD-MM-YYYY')
AND (#3# IS NULL OR b.loc IN csv[#3#])
ORDER BY UPPER(l.locld), UPPER(c.borcatld), b.joindate sortdate
```

Having changed the columns in the query, the Generate columns button must be used again to populate the column records. The change of the third column from a Date to a Text value will NULL the format given.

This will add a new column record for the sortdate. Specifying a % value of zero will prevent the values in this column from appearing in the report.

If there are any problems processing the query to generate or update the column records, the column records will remain unaffected. Should this occur, the error will be added in a comment at the top of the Query field.

Finally, the friendly format output of the parameter value is generated:

```
SELECT DISTINCT l.locld, c.borcatld,
   to_char(b.joindate, 'FMMonth, YYYY') showdate,
   COUNT(*) OVER (PARTITION BY b.loc, b.borcat,
```
Having changed the columns in the query (one column has been added), the Generate columns button must be used again to populate the new column record for “PARAMDISPLAY”. This should also be given a % value of zero.

Adding Parameters

The example query above handles 3 parameters as follows:

Start Join Date

- This is a single value, mandatory date field. The nature of the data likely indicates that Combo (mandatory) should be used together with the SQL to select the end of a holiday record
- The SQL should be adjusted, if necessary, to return the value in the DD-MM-YYYY format expected by the query specified above. For example:

```sql
SELECT KeyValuePair(to_char(holfin, 'DD-MM-YYYY'),
    holld||' '||l.locld||
    to_char(h.holfin, '" "("-Dy DD-Mon-YYYY")"'))
FROM holidays h, locs l
WHERE h.loc = l.loc(+)
ORDER BY h.holfin
```

- Note that this SQL has two date formats. The one to adjust is the first (in red) as this is used for the parameter value to pass to the SQL. The second date format is for display purposes when selecting the parameter value to use.
End Join Date

- This will be best configured like the Start Join Date parameter, but selecting the SQL for the start of a holiday record
- The SQL should be adjusted, if necessary, to return the value in the DD-MM-YYYY format expected by the query specified above

Locations

- This is an optional parameter for a repeatable value. It should be specified as a Multipick together with the SQL for choosing a Location.

**Preparing to Preview the Report**

Once the parameters have been configured, the Generate Layout method should be used to produce a suitable layout for this report. Once complete, it may be necessary to log out and log back in again to use the new layout. The message presented will advise accordingly.

When ready, display the report record in modify mode, specify all mandatory parameters and select a suitable output type before clicking the preview button.