Applying Best Practices, QA, and Tips and Tricks to Our Reports

If we had to summarize all we have learned so far, put it into a nutshell, and squeeze in just the very best of everything, this is how that kind of chapter would have looked like.

In this chapter, we will discuss best practices, tips and tricks, and how to check your report data integrity.

In this chapter, we will cover the following sections:

- Applying best practices to the reports
- The way of QA
- Three useful formulas

Applying best practices to the reports

This section will illustrate the application of best practices to the reports.

Getting ready

For every stage in the report development, we can apply best practices.

Best practices can be applied to the following stages:

- The query development
- The report formatting
- The Web Intelligence document

How to do it...

So, first of all, we will start with the query development stage; here, it is best to apply the following practices:

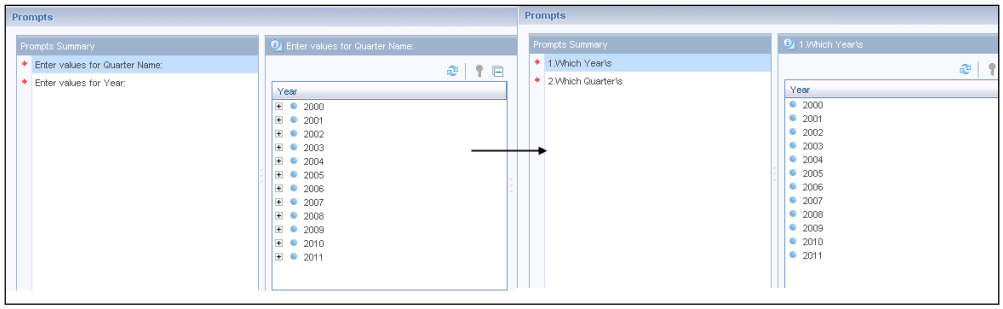
1. If possible, create your new report in the SAP BI development environment. Here, the runtime can be more suited for development purposes as well as the fact that there is usually low data traffic in the development environment. After creating the report, you can either use the BI team to transfer the report or use the **Web Rich Client**.
2. When you create the query, work with the same method: first, create the conditions of the query as it is the core of the query. Once the query filters are set, move on to the result objects.
3. When you drag-and-drop result objects, use the following order: dimensions first (character objects then dates), measures second. If you are also using attribute objects, drag them after the dimensions. In this way, it will be easier for you and for other developers to understand the query structure.

   There is no one way to order your objects, and certainly using a hierarchy-based order is good as well (Country | Area | City). What's most important is to apply those methods in all of your reports so that you can work with conventions that will make the report change management simple and edit actions to come.

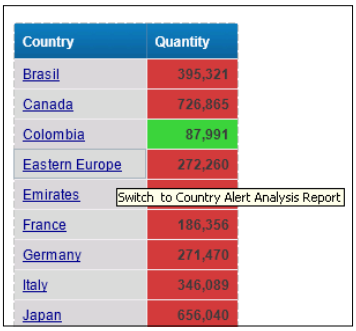
4. Try to use short numeric codes rather than long character objects in the query filter. Using code rather than characters will usually make the query work better and faster.
5. Use optional prompts rather than **ALL values** predefined prompt filters, although they are already built and can save some time in building the query; they use the OR logic that can make the query work harder.
6. Name the query/ies—even if you have just one query, it would be better to name the query. When you have several queries, it is easier to understand which query is doing what according to its name, and if one of the queries fails to run or doesn't retrieve data, then it's easier to diagnose the problem and locate the problematic query.
7. Use a query striping definition—this option can be found in the query panel and sends only those query objects to the database that are actually being used along the report tabs, displayed in tables, charts, and cells, or being part of a formula or a variable, which again is actually used in the report. This option is covered in *Chapter 2*, *Creating New Queries*.

Since most of the reports are being edited, changed, and adjusted for different audiences, we can come across reports that are very detailed, for example, but which actually display summarized data. In such cases, which can happen a lot, just the summarized data will be sent back to the database and by doing so, boosting the query performance.

8. If you require a certain order for the prompts to appear in, add a number abbreviation before the prompt's text; this will establish the right order as well as be clearer for the user as to how many prompts are required to answer, as shown in the following screenshot:

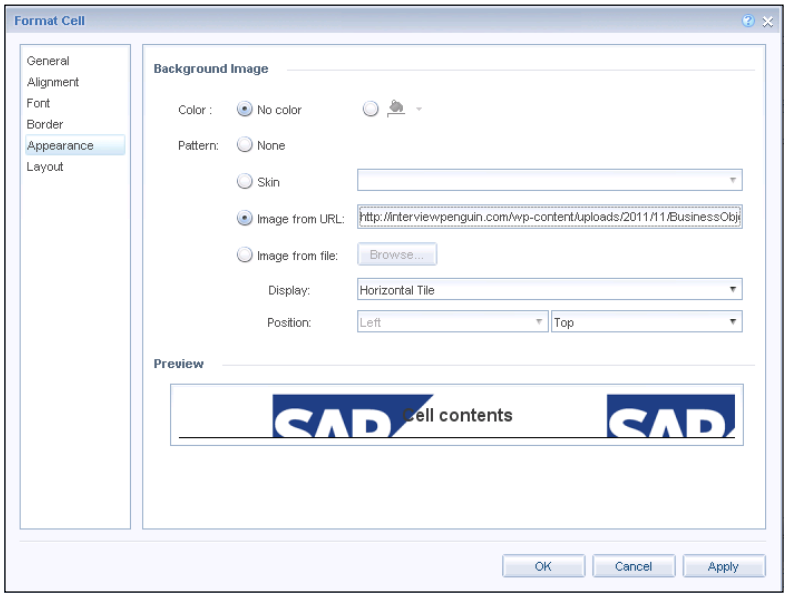


9. When you edit the query, use the **Search** option, and mark the result objects in order to locate the new objects you require or a relevant folder fast. The **Search** option is used to locate objects in the universe structure, mark a specific result object in order to find out its location under a certain universe folder.
10. Using conditional formatting plus plus using the **Document Link** option is one of the best combinations to trigger a detailed report for the right reasons (also called guided analysis), as shown in the following screenshot, and can help you reduce the number of unnecessary reports and large data volumes. We can activate a drill down report only for the values which are marked red and require further detailed analysis.



For more information on how to create conditional formatting, refer to *Chapter 6*, *Formatting Reports*, and on how to create report links, refer to *Chapter 10*, *Using Hyperlinks*.

11. You can use data from tables that are not connected directly in the universe by using subqueries and the **Union** query; depending on your business requirement, this can be a suitable solution.
12. For advanced report developers, using an outer join will make you pay in low performance and potentially long running time of the query; consider using another query (which will use a "Full Outer Join" logic), or in some cases, crosstab tables (see the *Working with cross tables* recipe in *Chapter 4*, *Working with Tables*).
13. When you create a new query, make sure to apply the changes; so, even if you don't run the query, you won't lose your query structure.
14. Use templates for quick formatting. Templates usually contain the basic report design already such as title, table colors, page sizing, and logos. Since the templates are not straightforward enough in Web Intelligence, you can use dummy reports that contain only the formatting. And before you start to develop your real report you will first access this template's empty data reports that contain only formatting.
15. When using logos in cells, use the picture URL of the public folder path as the logo can be changed, as shown in the following screenshot. It will be much easier to use the picture mapping rather than going through each of the reports and changing the picture locally.



An additional option is to ask for your SAP BI administrator to place the image in the image folder of the SAP BI4 server from which the reports can read pictures and use an even simpler syntax, which will be `boimg://logo.jpg`.

16. Give meaningful names to your report variables and delete all unnecessary variables.

17. Use a **Report Selection Criteria** tab; in this report tab lies the report purpose explanation. The prompts used are perhaps the queries used, the query structure, and other useful information such as remarks.
18. If you don't require the data, purge the report it will open faster, and in general, the entire disk storing the reports will save space.
19. Reports with prompts work better when the **Refresh on open** option is set since as soon as they open, the prompts are displayed, and it's very clear for the user that they are required to fill them in order to get the updated data.
20. Make sure to name all of the report tabs—same as naming the queries and it's easier to navigate through the report tabs.
21. When you make changes in a report, it's best to duplicate the report tab you are working on; in case anything goes wrong and you save the report, the original tab won't be affected.
22. Name the tables/charts and the cells you are using as well; when using the document structure and filters panel, it will help you navigate easily between the report elements.
23. Use concatenate formulas to eliminate unnecessary cells. This will help you reduce the number of cells and eliminate the ailment of several cells.
24. Use naming conversions for your reports; for example, if it is a finance department report, you can number it and use some kind of prefix such as `3.Finance_Report_name`; this is probably the most important step in order to create a well-organized and accessible repository.
25. Use the description and keyword properties of the report—the description should say what the purpose of the report is and who uses it.

    The keywords can be used for searching the reports; this is similar to tagging pictures or files.
26. Avoid report document inflation. Use prompt filters for different query filtering instead of a fixed filter with a specific value; if your report requires another time filter for scheduling, consult the universe designer for a smart universe-based solution.
27. Use the appropriate folder structure to store your reports and build folders according to the information areas you interact with and delete inactive reports.
28. In case you require different versions of the same report because of different time filters, add that distinguished filter to the report name, for example, `Monthly_Sales`.
29. Avoid report document inflation. If some business users require the same report but claim that they just need the data not the formatting, don't create another report, simply add another report tab with a "data dump" for easy download to Excel.

The way of QA

How can you know for sure that the results of your query are correct? What do you do in cases when the query doesn't retrieve results although the database is updated?

How can we QA our data? How can we track inconsistencies in our calculations and the report variables?

The entire purpose of QA is to find the problems and to distinguish whether the problem originates in the query level or if it happens in the report level.

In this section, I'll discuss how to use various methods of testing, checking our query, and report integrity.

Getting ready

As a common business user, we can't take responsibility for errors in the database or in the universe level, but we can sometimes track and find them.

The first ground rule is don't work without getting your IT support, especially the BI team support. If you aren't sure about the results and you can't use any of this recipe's QA methods, get the proper assistance from your BI team.

Ideally, every universe has a developer who is familiar with the universe structure as well as with the business area. When you have doubts or a lack of QA tools, consult your universe designer in order to get "first aid".

Still, there are a lot of things we can do before consulting the BI team.

How to do it...

In order to learn how to identify what went wrong in our report, we first need to get a basic business understanding of what is right (what counts as a valid transaction, how the net sales is calculated).

Business users can often have a good sense of the data, based on their business knowledge and their daily way of work and interaction with the operational systems (CRM, ERP, filesystems, daily data flows into e-mails).

That is of course not always the case, so what can we actually do?

In the query level, perform the following steps:

1. Make sure you are using the right objects. Business users can sometimes drag irrelevant objects when they are not that familiar with the universe structure; one of the most common mistakes is using the wrong date (for example, using the **Expiry Date** object instead of the **Estimated Expiry Date** object).
2. Use the preview first to check if you are getting reasonable rows.
3. The "no data to retrieve" message is not an error, it's an informative message; the most common and easiest way to check why your query didn't return any values is to check the values in the query filters—are they reasonable?
4. Elimination—when you don't have enough business or universe knowledge to determine what the reason is for the query to fail from retrieving rows, eliminate one condition at a time, rerun the query, and check if it's running now. This method will help you isolate the reason for no rows.

   In some cases, you will need to eliminate result objects since they can create joins and a structure that might not be valid for your expected query. Remember that errors can be created by the universe designer as well because of the structures that are still not supported in the universe.

   If after the elimination process you still don't get any results, the problem most likely lies in the universe or at the database level.

5. Comparing source to target—we are definitely not BI developers, but if we had eight new customers to our business and we expect to see them in our **New Customers** report, simply make sure that you get all of them (the records) in your report.
6. If you have some SQL skills, then you can run the same SQL script directly against the database using some sort of SQL tool, without going through the universe. This is a simple comparison I use sometimes to make sure there is no data reduction caused by the reporting tool (Web Intelligence).
7. Make sure your universe is mapped against the relevant database (this information can be supplied by the universe developer).
8. Are you sure the data is updated? Are you getting daily updates that the data has been successfully updated by the BI department?

   A very known case in the ETL process is that sometimes one of the tables might not be updated successfully, which of course affects the universe and the reports using it. Getting notifications about failed data processes can save lots of time and effort trying to figure out why we are missing data on one of our reports.

9. Use QA queries—check your main report data by using another detailed query (very close to a "data dump" query) that will check the data both in a detailed fashion as well as the measure consistency across detailed levels.

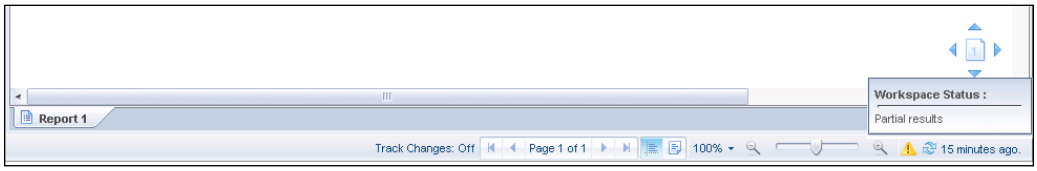
   A very common QA scenario would be running a detailed report using only a single customer name prompt, getting well-detailed and unrestrictive data about the customer, and then comparing it to your main report. For example, in this way, you may find out that a particular customer would always access the database as a new customer and left although you are looking for fresh customers without that type of history.

   This type of testing should be executed at least several times; it's hard to determine the exact number of times since there are many QA approaches, but a minimum of 10 different tests is a good start.

10. When comparing an ERP/CRM system report to a universe report, are they using the same logic, filters, and so on? This is one of the tests for which you most likely require IT support.

    Take in concern that more the data warehouse or the data model is newer, we can expect more data problems.

11. Are you getting relevant as well as irrelevant data in your report? Check your elimination query filter logic: are you using not in list, not equal to logic or a subquery, using not in, or a minus query?
12. Another point for skilled SQL users—does the SQL being generated make sense? Are the relevant tables and joins being generated?
13. One of the most important things to check—does the universe/query have a max row or max retrieval time limit? Many times, I have come across a forgotten query that was changed several times by different users where one of them limits the query results to a fixed number of rows, probably for testing purposes.
14. Besides the query properties panel where you can actually view those definitions, in case the query does use those limits and it hasn't accrued our mind, then pay attention to the bottom-right bar displaying a warning notification about partial results in yellow, as shown in the following screenshot:



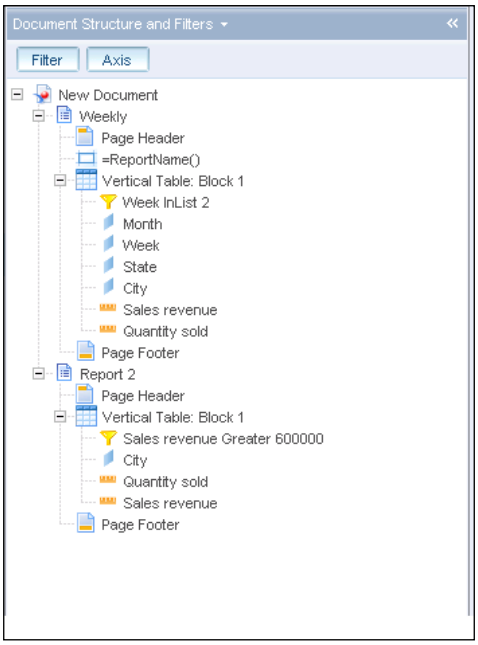
At the report level, perform the following steps:

1. Check the report filters—filters can affect the final result of measures, missing values, and other divergences. When I get a report from a business user who claims they are missing some rows, I always check the filters first.

   Whenever a user creates a filter and for whatever reason forgets about its existence, it can cause difficulties in understanding why they are facing a certain result.

   The easiest way to get a global view of the entire filters in the report is to use the predefined cell **Report Filter summary** that will display the filter information about filters in the report in a single cell.

   Another way to explore which filters are affecting our data would be to navigate to the **Document Structure and Filters** option, as shown in the following screenshot:



   On this exception, remember that the table has certain properties in the general category that can affect the display of certain types of rows.

2. Check the data amount—before I start testing a report in depth, I always check the number of rows retrieved first by using the data view (described in the *Navigating between the view modes* recipe in *Chapter 3*, *Working inside the Report*), this will give you the most basic understanding of why we are dealing with the same scale of rows, whenever an SQL script is ran against the database by using an SQL tool and returns a certain amount of rows; comparing that amount to our report is a very good start to check the data integrity.

3. Merging data providers—as we have seen in *Chapter 8*, *Merging Data*, merging data providers can affect the data display as well, especially when using the left or the right merge options. A basic test would be to check the tables before and after the merge and look for missing rows or changes in the measures values.
4. Check for hidden dimension objects that may affect the aggregation level of the table in order to get a better understanding of the calculations results behavior.
5. Measure values—probably the most important and interesting subject to explore and test; what can we expect to go wrong? Values too high or too low and values not being summed correctly.

   What should we check and how? The most basic approach should be are we getting the total correctly regardless of any table formatting, report filter, or any other kind of specific table aggregation? If the answer is yes, then the problem doesn't lie in the query level, but probably in the report level.

   It is important to take into account that although the total is ok, there can still be a row level measure problem.

   What's next? When the values are too high or too low, usually the problem lies in the query structure or in the universe structure.

   What reasons can cause an inconsistency in the numbers?

   - Using the wrong tables together in the query (with unbalanced aggregative levels, for example, a monthly table with a daily table)
   - Using a wrong date in the query filters
   - Using tables with the wrong join between them
   - Using data from unmerged queries together
   - A wrong aggregation projection definition of the measure object in the universe (for example, max instead of sum)

   The "object by object" elimination method is still a great way to identify whether there is a specific object causing the numbers to drastically change; locating the right object can help the universe designer to take it from there and solve the problem.

   Another important check in this note would be aggregating the measure with different levels of dimensions and making sure we are still getting the same totals.

   Under this type of check there are the exception measures such as averages and client count as there can be reoccurrences.

   For example, how many clients we have in our report is one calculation where its result can be different from how many customers bought a specific product. Since the same customer could have bought several products, they can be counted several times in the product level and we will probably get a higher customer count in the total.

6. Wrong context of the measure—as we discussed in the *Using extended syntax* recipe in *Chapter 9*, *Using Formulas and Variables*, if we don't define the appropriate context, we might get an "out of context" calculation. Applying the right context will fix it.
7. Wrong calculation logic—in some cases, we have a complex calculation combined from several parts where one of them is calculated wrong. Testing the calculation parts, one by one, will help us trace the problematic calculation part (this is also why it is so important to structure complex variables from subvariables).

   Another good example for a famous calculation being calculated wrong is the weighted average. Sometimes, you just need to know a bit of math!

   You can read about how to calculate the weighted average correctly on `http://en.wikipedia.org/wiki/Weighted_arithmetic_mean`.
8. Duplicate rows—there is no such thing as duplicate rows. A duplicate row is a row such that all its columns' values are identical to the next/former row.

   By default, duplicate rows aren't displayed in Web Intelligence tables, but you can find in the table properties the **Avoid duplicate row aggregation** option, which is an option we can apply especially when we don't use measures that perform row aggregation at the query level (group by of the rows).

   Duplicate rows is more a wrong concept rather than something that actually happens; it usually occurs by adding an irrelevant dimension which isn't duplicating the row but rather adding rows due to its extra level of details.

   I have come across several times requests to display "all the rows" for counting occurrence reasons, and data governance, but these kind of requirements can be easily eliminated with smart data warehousing modeling and usually pointing on a premature data warehouse.

There's more...

Designing QA and test scenarios for our reports is one of the most important things in the lifecycle of any report development process.

Some of the practices mentioned require skills; some require the help of the IT or the BI team, some are based on the assumption that you get to a level you are professional enough as well as you have a basic business sense what is wrong and what is right.

My recommendation is try to work as close as possible with the BI team.

Report on bugs you find, ask for their assistance in checking the report, meet with the universe developer on a recurrent basis, and make sure you stay in the loop with updates and new changes.

Remember that the universe is built for business users so that they can develop their own reports and that many times a universe takes its developer course because of an analyst, a power user, or a business user that spotted some problem nobody else saw prior to him.

Three useful formulas

Tips and tricks is an endless topic that can easily fill a book or two. Among the various types of tricks I choose to focus on are the most useful ones, the ones you will probably need to use in your reports.

Getting ready

Whenever we are using several prompts in a query, we will rather display all the prompts in one cell (we do have a predefined cell, but its formatting is fixed and comes with additional characters which aren't suited for everyone).

Let's see how we can create a formula that will concatenate all the prompts as well as show each prompt in a separate row.

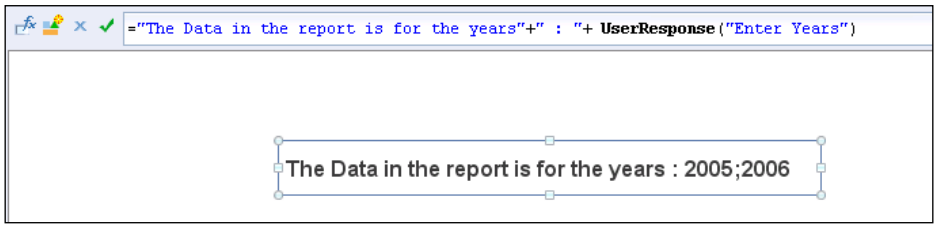
How to do it...

We have two prompts in our report: a **Year** and a **Quarter**.

The first part of the formula will concatenate on explanatory text to the **Year** prompt values and will use the following syntax:

```
="The Data in the report is for the years"+" : "+
UserResponse("Enter Years")
```

We will get the following result:



Now, we will add the second prompt, but in order to make it appear in the next line, we will add the char (`10`) or char (`13`) logic.

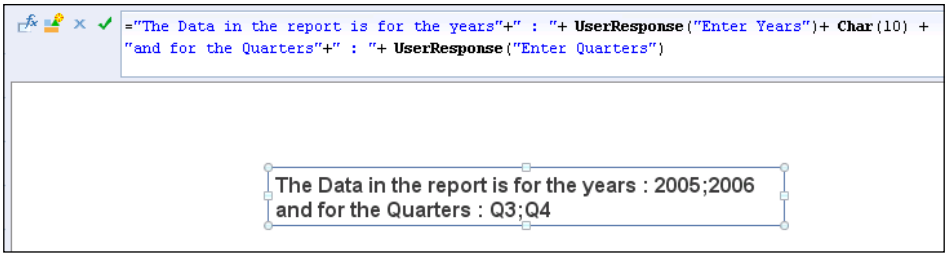
Both of these character numbers are feed lines, that is, a carriage return such as *Ctrl* + *Enter*.

Using this simple function between every prompt we want to concatenate will push them one line below.

Our new perfected syntax will now look like this:

```
="The Data in the report is for the years"+" : "+ UserResponse("Enter
Years")+ Char(10) +
"and for the Quarters"+" : "+ UserResponse("Enter Quarters")
```

And our cell will look like this:



That was simple enough. This formula can be added to a template report, to be used upon the rest of our reports; the only thing we require to change is the text of the prompts according to the query we are using.

Our second formula is very common yet still very handy.

In some cases, our formula uses a deviation that might use `0` as its divisor; in rows where the divisor is `0`, we will get the `#DIV/0` error, as shown in the following screenshot:

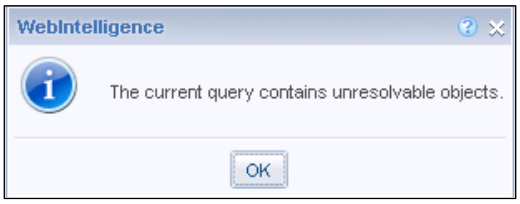
Sales revenue
$2,293
$2,058
#DIV/0
#DIV/0
$2,343

In order to suppress this error message, we will use the `IsError()` formula located in the logical functions folder in the formula editor inside our formula. The syntax will be `=If IsError([Our object])Then 0 Else [Our object]`.

The result will be the following:

Sales revenue	Fixed formula
$2,293	$2,293
$2,058	$2,058
#DIV/0	$0
#DIV/0	$0
$2,343	$2,343
$225	$225

Our third formula is an advanced formula for situations where we are getting the following message:



This message will occur whenever an object or objects which are part of your exiting query have been deleted for some reason from the universe.

The problem is that we can't know which objects were removed because as soon as we go to the query panel, the query changes as well as the SQL code.

In order to get the SQL code before it changes due to our query panel visit, we will use the following formula in a cell: `=DataProviderSQL(DataProvider([any query object]))`.

Based on this formula, we will now be able to view the SQL code and understand which object is the problematic one by comparing this script to the one the query pane generates: