

# Standards for Custom Reports with SAP SuccessFactors Learning Report Designer


CONFIDENTIAL





## TABLE OF CONTENTS

HANDLING DATES AND DATETIME CONVERSIONS .....	3
SCRIPT GUIDELINES .....	3
Use Different Alias Names for Tables .....	3
Add Table Alias Names to Columns Used in JOIN Conditions .....	4
Do Not Use WITH Clause inside FROM and JOIN Condition .....	4
Handling Connect by Queries - Example 1: .....	6
Handling Connect by Queries - Example 2: .....	7
Do Not Use LISTAGG .....	7
Do Not Use DBMS_LOB .....	8
Do Not Use Inner Keyword .....	8
Do Not Use WM_CONCAT .....	8
Do Not Use UNIQUE keyword .....	8
Do Not Use LOCATION Keyword as Alias .....	9
Do Not Use INSTR2 .....	9
Do Not Put Spaces in Logical Operators .....	9
Do Not Use Parentheses for Group By Clauses with Multiple Columns .....	9
Do Not Select All Columns .....	10
Use NVARCHAR Instead of varchar2 in Cast Functions .....	10
Aliases with Special Characters .....	10
Do Not Use Columns Names with Table Names .....	10
Use Consistent Data Types between DB Column Type and Report Designer Column Type .....	11
Do Not Use Case Statements with Mixed Datatypes .....	11
Using Keywords LIMIT and CURRENT_DATE as Aliases .....	11
Column Names in rptdesign file .....	11
Do Not Use '&' in Column Alias .....	11



We continue to optimize the performance and utilization of the SAP SuccessFactors Learning Report Designer. As such, ANSI SQL is our mandatory standard for SAP SuccessFactors Learning Report Designer. When you write report SQL in ANSI SQL, we have a higher confidence that those reports will deliver accurate and consistent results. Dates and datetime conversions may require changes in your reports. Please review guidance below on handling dates and datetime conversion.

## HANDLING DATES AND DATETIME CONVERSIONS

Time zone conversion can be difficult when you write a custom report in Learning. In general, please follow this standard: start with UTC and convert the date-times into the correct time zone for the user. This applies to *date-time* fields:

- *Date-time* fields (as opposed to date fields) appear to report readers in UTC by default.
  - If you apply explicit BIRT time zone conversion to a date-time field, the report honors the conversion: readers of the reports see the date-time converted.
  - Do *not* for example, assume that you can rely on the time zone of the data center. Instead, assume that the raw date-time is in UTC.
- *Date* fields (as opposed to date-time fields) should *not* have a time zone conversion
  - For example, assignment date is a *date* field, so it does not require time zone conversion.
  - Do not use the BIRT operator for time zone conversion on *date* fields because time zones make sense for *date-time* fields only.

## SCRIPT GUIDELINES

Follow these guidelines when writing SQL script for custom reports.

### Use Different Alias Names for Tables

Do not use the same table alias name for more than one table.

#### *Previous Script*

```
SELECT DISTINCT c.cpnt_typ_id,
c.cpnt_id,
c.rev_dte ,
cs.cpnt_src_desc,
from pa_cpnt,
      pa_cpnt_src cs,
      pa_cpnt_subj cs,
      pa_cpnt_formula cf
where cf.cpnt_typ_id (+) = c.cpnt_typ_id
and   cs.cpnt_id(+)   = c.cpnt_id
```

#### *Standard Script*

```
SELECT DISTINCT c.cpnt_typ_id,
c.cpnt_id,
c.rev_dte ,
cs.cpnt_src_desc,
from pa_cpnt,
      pa_cpnt_src cs,
```

```

    pa_cpnt_subj cs ,
    pa_cpnt_formula cf
where c.cpnt_typ_id = cf.cpnt_typ_id (+)
and   c.cpnt_id     = cs.cpnt_id(+)

```

### Add Table Alias Names to Columns Used in JOIN Conditions

Add a table name alias to a column used in join condition. Column names without an alias are not added in the join clause condition.

#### *Previous Script*

```

Select
pr.FORMULA,
pr.CURRENCY_CODE
From PA_CPNT_FORMULA pr
Where IS_DEFAULT (+) = 'Y'
and   fin_var_id (+) = 'ItemDefaultPublishedPrice'

```

#### *Standard Script*

```

Select
pr.FORMULA,
pr.CURRENCY_CODE
From PA_CPNT_FORMULA pr
Where pr.IS_DEFAULT (+) = 'Y'
and   pr.fin_var_id (+) = 'ItemDefaultPublishedPrice'

```

### Do Not Use WITH Clause inside FROM and JOIN Condition

Use a SELECT statement in place of a WITH clause when you are inside a FROM and JOIN condition. Use WHERE to add additional SELECT statements.

#### *Previous Script*

```

SELECT *
FROM
(SELECT *
FROM
(WITH summary AS
(SELECT
sct.*,ROW_NUMBER() OVER(PARTITION BY
sct.sc_stud_id_not_used,sct.CPNT_ID,sct.REV_DTE
ORDER BY sct.req_dte,sct.assgn_dte ASC) AS rk
FROM (SELECT NULL as compl_dte,
NULL as compl_stat_id,
NULL as compl_stat_desc,sc.cpnt_typ_id,
sc.cpnt_id,
sc.rev_dte,
sc.rtyp_id,
sc.stud_id AS sc_stud_id_not_used,
sc.assgn_dte,
sc.req_dte,
sc.exp_dte

```

```

FROM pa_stud_qual_cpnt sc
WHERE (sc.compl_dte IS NULL
OR sc.exp_dte < CURRENT_DATE
OR sc.retrng_int > 0)
)sct)
SELECT s.*
FROM summary s WHERE s.rk=1)
)

```

### **Standard Script**

```

SELECT *
FROM
(SELECT *
FROM
(SELECT * FROM
(SELECT sct.*,ROW_NUMBER() OVER(PARTITION BY
sct.sc_stud_id_not_used,sct.CPNT_ID,sct.REV_DTE
ORDER BY sct.req_dte,sct.assgn_dte ASC) AS rk
FROM (SELECT NULL as compl_dte,
NULL as cml_stat_id,
NULL as cml_stat_desc,
sc.cpnt_typ_id,
sc.cpnt_id,
sc.rev_dte,
sc.rtyp_id,
sc.stud_id AS sc_stud_id_not_used,
sc.assgn_dte,
sc.req_dte,
sc.exp_dte
FROM pa_stud_qual_cpnt sc
WHERE (sc.compl_dte IS NULL
OR sc.exp_dte < CURRENT_DATE
OR sc.retrng_int > 0)
UNION
SELECT
NULL as compl_dte,
NULL as cml_stat_id,
NULL as cml_stat_desc,
sc.cpnt_typ_id,
sc.cpnt_id,
sc.rev_dte,
sc.rtyp_id,
sc.stud_id AS sc_stud_id_not_used,
sc.assgn_dte,
sc.req_dte,
NULL AS exp_dte
FROM PV_STUD_COURSE sc
WHERE sc.compl_dte IS NULL
)sct)
WHERE rk=1
))

```

### **Do Not Use Oracle Specific Tuning Statements**

Examples include:

```

SET ARRAYSIZE
SET LINESIZE

```

## Handling Connect by Queries - Example 1:

### Previous Script

```
SELECT

'BOR' as BOR,UPPER(s.STUD_ID) AS eID,LOWER(s.EMAIL_ADDR) AS Email,s.FNAME AS First_Name,s.LNAME AS
Last_Name,tEmployee.USER_VALUE AS Employee_Number,tOrg.ORG_DESC AS Department,tJob.JP_DESC AS
Title,s.Super,s.JP_ID as Job_Code,us.user_desc as Manager_Level_Desc,su.user_value as
Manager_Level_Code,s.dmn_id,s.org_id,tCompany.USER_VALUE AS Company,TO_CHAR(s.Hire_dte, 'YYYY-MM-
DD') as Hire_Date,s.JL_ID as Job_Location_Id,tLoc.JL_DESC AS
Job_Location_Desc,LEVEL,DECODE(s.NOTACTIVE, 'N', 1, 0) as Active

FROM PA_STUDENT sLEFT JOIN PA_ORG tOrg
on s.ORG_ID = tOrg.ORG_ID
LEFT JOIN PA_STUD_USER tEmployee
on s.STUD_ID = tEmployee.STUD_IDAND tEmployee.COL_NUM = 1
LEFT JOIN PA_JOB_POS tJob
ON s.JP_ID = tJob.JP_ID
LEFT JOIN PA_STUDENT tMgr
On s.SUPER = tMgr.STUD_ID
LEFT JOIN PA_STUD_USER su
ON s.STUD_ID = su.STUD_IDAND su.col_num = 2
LEFT JOIN PA_USRRF_STUD us
ON su.user_value = us.user_idand us.col_num = 2
LEFT JOIN PA_STUD_USER tCompany
on s.STUD_ID = tCompany.STUD_IDAND tCompany.COL_NUM = 4
LEFT JOIN PA_JOB_LOC tLoc
ON s.JL_ID = tLoc.JL_ID
WHERE s.SUPER != 'PLATEAU'

-- AND s.EMAIL_ADDR is not null-- AND s.NOTACTIVE = 'N'AND LOWER(s.EMAIL_ADDR) NOT IN
('conversion@invalid.com', 'invalid@capitalone.com')/**and [security:PA_STUDENT s]/START WITH
UPPER(s.STUD_ID) = 'OJG777'CONNECT BY PRIOR UPPER(s.STUD_ID) = UPPER(s.SUPER)ORDER SIBLINGS BY
s.LNAME
```

### Standard Script

```
SELECT *
FROM
(SELECT *
FROM
(SELECT * FROM
(SELECT sct.*,ROW_NUMBER() OVER(PARTITION BY
sct.sc_stud_id_not_used,sct.CPNT_ID,sct.REV_DTE
ORDER BY sct.req_dte,sct.assgn_dte ASC) AS rk
FROM (SELECT NULL as compl_dte,
NULL as compl_stat_id,
NULL as compl_stat_desc,
sc.cpnt_typ_id,
sc.cpnt_id,
sc.rev_dte,
sc.rtyp_id,
sc.stud_id AS sc_stud_id_not_used,
sc.assgn_dte,
sc.req_dte,
sc.exp_dte
```

```

FROM pa_stud_qual_cpnt sc
WHERE (sc.compl_dte IS NULL
OR sc.exp_dte < CURRENT_DATE
OR sc.retrng_int > 0)

UNION
SELECT
NULL as compl_dte,
NULL as compl_stat_id,
NULL as compl_stat_desc,
sc.cpnt_typ_id,
sc.cpnt_id,
sc.rev_dte,
sc.rtyp_id,
sc.stud_id AS sc_stud_id_not_used,
sc.assgn_dte,
sc.req_dte,
NULL AS exp_dte
FROM PV_STUD_COURSE sc
WHERE sc.compl_dte IS NULL
)sct)
WHERE rk=1
))

```

## Handling Connect by Queries - Example 2:

### *Previous Script*

```

SELECT
level l,s.stud_id,s.super,sys_connect_by_path(stud_id,' ; ') AS manager_hierarchyFROMpa_student s
CONNECT BY
PRIOR s.stud_id = s.super

```

### *Standard Script*

```

SELECT
stud.level l,s.stud_id,s.super,replace(stud.path,'/',';') AS manager_hierarchy
FROM
pa_student s,pa_student_super_hview stud
WHERE
s.stud_id = stud.result_node

```

## Do Not Use LISTAGG

Use STRING\_AGG instead.


### *Previous Script*

```

LISTAGG(ap.ap_desc, ' | ') WITHIN GROUP( ORDER BY ap.ap_desc) FROM pa_assgn_prfl ap,
pa_stud_assgn_prfl sap WHERE 1 = 1 AND ap.ap_id = sap.ap_id AND sap.stud_id = 'stud101'

```

### *Standard Script*



```
STRING_AGG(ap.ap_desc, ' | ' ORDER BY ap.ap_desc) from PA_ASSGN_PRFL ap , PA_STUD_ASSGN_PRFL sap
where 1=1 and ap.ap_id = sap.ap_id and sap.stud_id = 'stud123'
```

### **Do Not Use DBMS\_LOB**

Use SUBSTR instead.

#### ***Previous Script***

```
SELECT
dbms_lob.substr(cbt.activity_tree,4000,1) activity_tree from pa_cbt_cpnt cbt where
dbms_lob.substr(cbt.activity_tree,4000,1) IS NOT NULL
```

#### ***Standard Script***

```
substr(cbt.activity_tree,4000,1) activity_tree from pa_cbt_cpnt cbt where
substr(cbt.activity_tree,4000,1) IS NOT NULL
```

### **Do Not Use Inner Keyword**

Inner is a reserved keyword.

#### ***Previous Script***

```
SELECT inner.* --,
SUBSTR(status_remday, 1,1) AS COMPLETE, -- SUBSTR(status_remday, INSTR(status_remday,'|',1)+1)
```

#### ***Standard Script***

```
SELECT inner1.* --,
SUBSTR(status_remday, 1,1) AS COMPLETE, -- SUBSTR(status_remday, INSTR(status_remday,'|',1)+1)
```

### **Do Not Use WM\_CONCAT**

Use STRING\_AGG instead.

#### ***Previous Script***

```
select WM_CONCAT (dept_name, ' , ' ) from department
```

#### ***Standard Script***

```
select STRING_AGG(dept_name, ' , ' ) from department
```

### **Do Not Use UNIQUE keyword**

Use DISTINCT instead.





### ***Previous Script***

```
SELECT UNIQUE stud_id FROM pa_student ;
```

### ***Standard Script***

```
SELECT DISTINCT stud_id FROM pa_student ;
```

### **Do Not Use LOCATION Keyword as Alias**

Use another string as the alias.

### ***Previous Script***

```
select stud_id from pa_student location;
```

### ***Standard Script***

```
select stud_id from pa_student my_alias;
```

### **Do Not Use INSTR2**

Use INSTR instead.

### ***Previous Script***

```
select instr2('demostring','d',1,1) from dual;
```

### ***Standard Script***

```
select instr('demostring','d',1,1) from dummy;
```

### **Do Not Put Spaces in Logical Operators**

For example, if you have != or <=, do not put a space between ! and =

### ***Previous Script***

```
SELECT * from pa_student where pa_stud ! = 'AA';
```

### ***Standard Script***

```
SELECT * from pa_student where pa_stud != 'AA';
```

### **Do Not Use Parentheses for Group By Clauses with Multiple Columns**

### ***Previous Script***

```
select stud.stud_id,stud.fname from pa_student stud group by (stud.stud_id, stud.fname)
```



### ***Standard Script***

```
select stud.stud_id,stud.fname from pa_student stud group by stud.stud_id, stud.fname
```

### **Do Not Select All Columns**

Select only the required columns.

### ***Previous Script***

```
select * from pa_student
```

### ***Standard Script***

```
select stud_id from pa_student
```

### **Use NVARCHAR Instead of varchar2 in Cast Functions**

### ***Previous Script***

```
select cast(job_id as varchar2(4000) from dual;
```

### ***Standard Script***

```
select cast(job_id as NVARCHAR(4000) from dummy;
```

### **Aliases with Special Characters**

Use double quotation marks around a column alias with special characters.

### ***Previous Script***

```
select su.user_value from pa_stud_user su where col_num = 130 and c.stud_id = su.stud_id) as  
Chaveárquico;
```

### ***Standard Script***

```
Select su.user_value from pa_stud_user su where col_num = 130 and c.stud_id = su.stud_id) as  
"CHAVEÁRQUICO";
```

### **Do Not Use Columns Names with Table Names**

Avoid using a table name with column names in SELECT statements. It adds unwanted schema name in the alias.

For example, do not use the following:

```
SELECT PA_INST.FNAME from PA_INST
```



## Use Consistent Data Types between DB Column Type and Report Designer Column Type

Note the DATE field in the bounded column and the DATE-TIME in result set columns within RPTDESIGN.

### Do Not Use Case Statements with Mixed Datatypes.

Use the `to_char` function in cases with mixed data types. For example, if the case statement resulting in a single column has integer and string values.

#### *Previous Script*

```
select decode(eq.ATTEMPT_LIMIT, '0', 'Unlimited', eq.ATTEMPT_LIMIT) as ATTEMPT_LIMITS from PA_EQB_QUIZ eq
```

#### *Standard Script*

```
select * from (select (case when eq.ATTEMPT_LIMIT='0' then 'Unlimited' else to_char(eq.ATTEMPT_LIMIT) end) as ATTEMPT_LIMITS from PA_EQB_QUIZ eq )
```

## Using Keywords LIMIT and CURRENT\_DATE as Aliases

Use double quotation marks when using keywords like LIMIT and CURRENT\_DATE as aliases.

#### *Previous Script*

```
select LIMIT from PA_TRAINING_REQUEST_LIMIT;
```

#### *Standard Script*

```
select "LIMIT" from PA_TRAINING_REQUEST_LIMIT;
```

## Column Names in rptdesign file

Use upper case for column names in rptdesign file while mapping.

### Do Not Use '&' in Column Alias

Do not use '&' in column alias in SQL and in rptdesign file.

[www.sap.com/contactsap](http://www.sap.com/contactsap)

© 2018 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies. See <http://www.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.