



# Introduction to CDS Views

## 1. Introduction

- CDS stands for Core Data Services.
- It is also based on Top down approach { i.e. Code to data paradigm } .
- It is an extension of Open SQL in ABAP.
  1. Data Definition Language is extended
  2. Data Query Language has been extended
  3. Data Expression Language has been extended
  4. Data control Language has been extended
- CDS are semantically rich data models in SAP S/4 HANA.
  - Semantically rich data means, We can use annotations in CDS Views which intern drives the functionality.
- We can build CDS views if our company is not using the HANA DB, but it may not give the high performance, So it is recommended to use HANA DB.
- Similar to AMDP CDS Views follows the code pushdown approach, However CDS Views always returns the single result set where as AMDP can return multiple result sets.

---

## **2. Difference between DDIC Views and CDS Views**

### **1. DDIC Views**

- DDIC Views cannot perform calculations, aggregations, grouping etc.
- DDIC View supports limited joins.
- DDIC View does not support nested views { i.e. view on view }
- In DDIC View, DB View consumption takes place at Application Server level.
- We can create DDIC View from SE11.

### **2. CDS Views**

- CDS Views can perform calculations, aggregations, grouping etc.
- CDS has vast support for joins and unions.
- CDS Views supports nested views.
- CDS View is executed at HANA Database Level { and we will be able to leverage the HANA database features }.
  - All the calculations will be performed at the HANA DB level.
- CDS View can only be created on Eclipse or Hana Studio.



## Creating CDS View

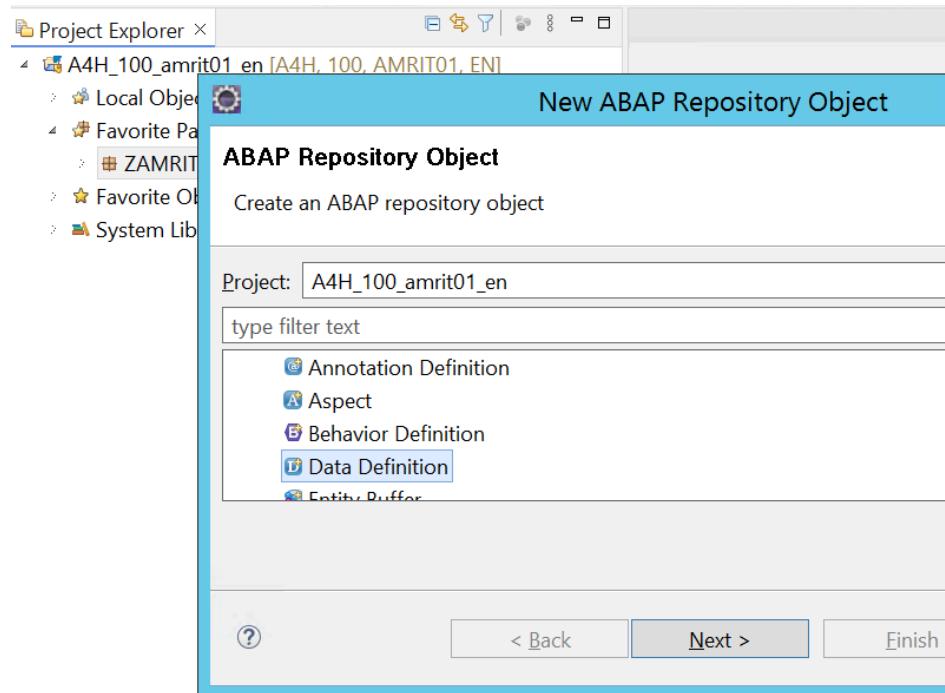
- Whenever we create a CDS View in ADT, A corresponding DDIC View is created into the SAP System.

### Requirement

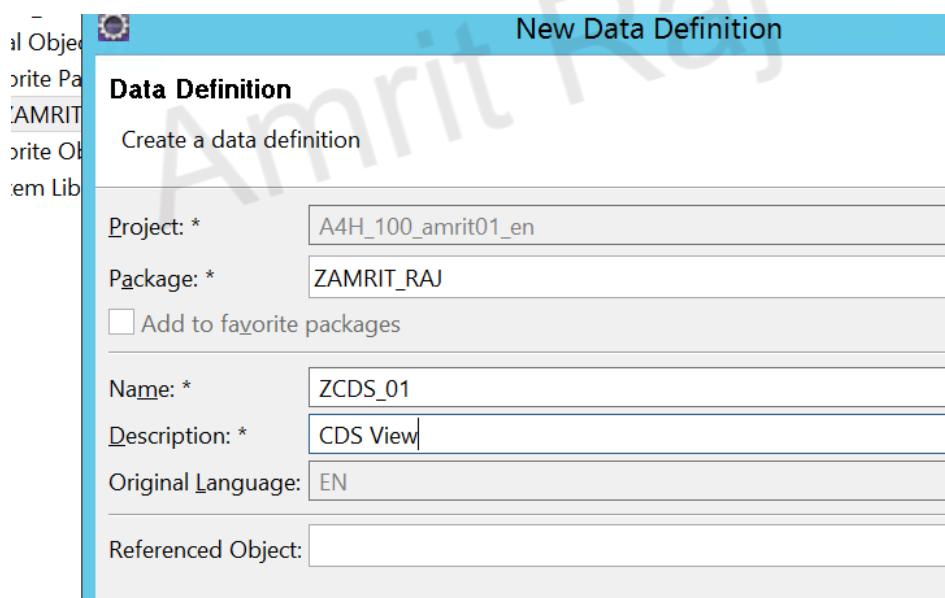
- Display the Material Master data using the CDS View.

### Implementation

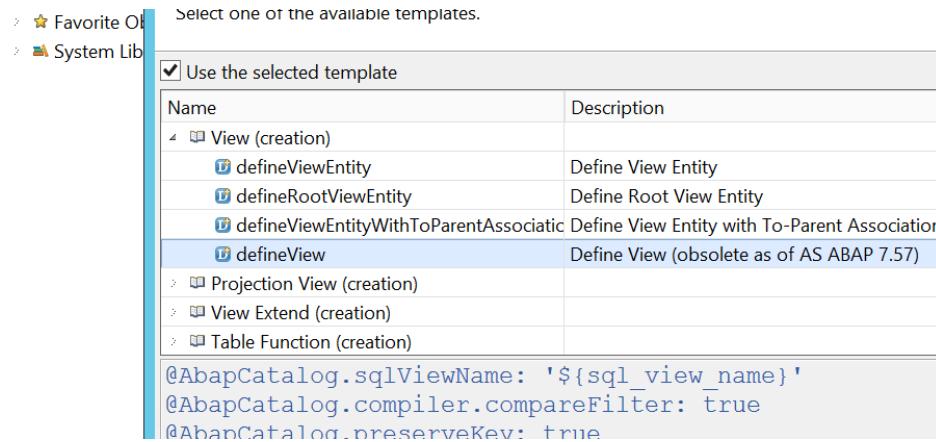
- Step 1 :- Right click on the package → New → Data Definition



- Step 2 :- Pass the name of the CDS Views and short description.



- Step 3 :- Assign the transport request.
- Step 4 :- Select one of the templates, first let's go with define view.



- Click on Finish.

```

@AbapCatalog.sqlViewName: ''
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'CDS View'
@Metadata.ignorePropagatedAnnotations: true
define view ZCDS_01 as select from data source name
{
}

```

- Step 5 :- Write the logic.

```

@AbapCatalog.sqlViewName: 'ZDDIC_01'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'CDS View'
@Metadata.ignorePropagatedAnnotations: true
define view ZCDS_01 as select from mara
{
    matnr as MATERIAL_NUMBER,
    case mtart
        when 'ROH' then 'Raw Materials'
}

```

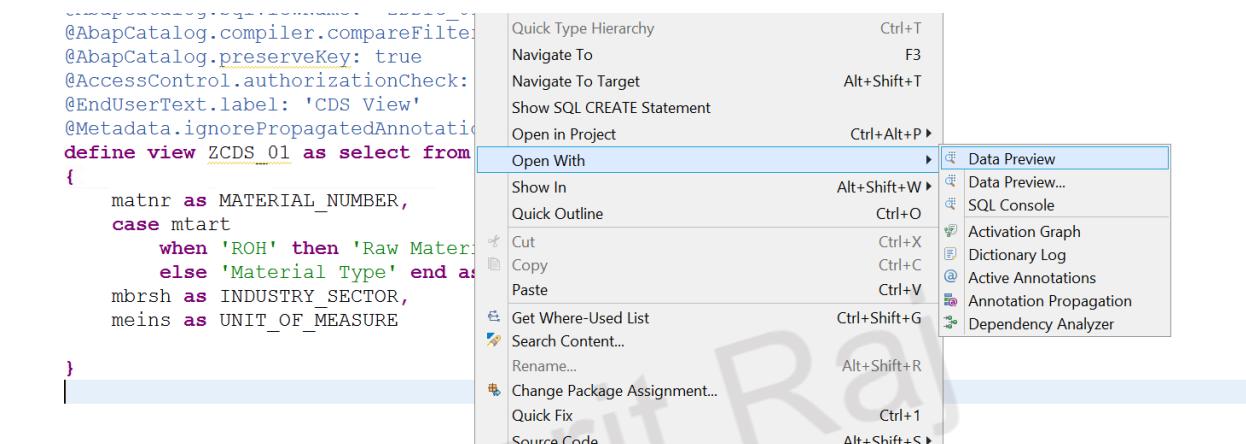
```

        else 'Material Type' end as Material_Type,
mbrsh as INDUSTRY_SECTOR,
meins as UNIT_OF_MEASURE

}

```

- Activate it.
- Step 6 :- Right click → Open with data preview.



## Note

- A DDIC View would have been created, you can check from SE11.

## Dictionary: Display View



CDS Database View ZDDIC\_01 Active

Short Description CDS View

Data Definition ZCDS\_01

Attributes Table/Join Conditions View Fields Selection Conditions Maint.Status



| View field      | Table          | Field              |
|-----------------|----------------|--------------------|
| MANDT           | MARA           | MANDT              |
| MATERIAL_NUMBER | MARA           | MATNR              |
| MATERIAL_TYPE   | DDDDLCHARTYPES | CHAR*000013*000000 |
| INDUSTRY_SECTOR | MARA           | MBRSH              |
| UNIT_OF_MEASURE | MARA           | MEINS              |

⚠ Generated DDL SQL views are only supported in limited way by SE11





# Consume CDS Views in ABAP Program

- We can use CDS Views as a data source for writing Open SQL Queries in ABAP Program.
- Using the CDS Views, we can store the data into the internal table and perform various kind of operations.

## Implementation

- Create a ABAP Program and write a select query using CDS View as the data source.

```
SELECT * FROM ZAR_CDS_VIEW1  
INTO TABLE @DATA(LT_TABLE).
```

TRY.

```
CL_SALV_TABLE→FACTORY(  
  EXPORTING  
    LIST_DISPLAY = IF_SALV_C_BOOL_SAP→FALSE " ALV Displayed in List Mode  
  *   R_CONTAINER =           " Abstract Container for GUI Controls
```

```

*  CONTAINER_NAME =
IMPORTING
  R_SALV_TABLE = DATA(LO_ALV)          " Basis Class Simple ALV Ta
CHANGING
  T_TABLE     = LT_TABLE
).
CATCH CX_SALV_MSG. " ALV: General Error Class with Message
ENDTRY.
LO_ALV→DISPLAY( ).
```

## Output

| Material |              | I... | Unit |
|----------|--------------|------|------|
| RM20     | Raw Material | M    | PC   |
| SG25     | AMRIT RAJ    | M    | PC   |
| SG23     | AMRIT RAJ    | M    | PC   |
| RM128    | Raw Material | M    | PC   |
| SG22     | AMRIT RAJ    | M    | PC   |
| RM122    | Raw Material | M    | PC   |
| SG124    | AMRIT RAJ    | M    | PC   |
| RM16     | Raw Material | M    | PC   |
| RM124    | Raw Material | M    | PC   |
| FG130    | AMRIT RAJ    | M    | PC   |
| SC30     | AMRIT RAJ    | M    | PC   |



# Creating CDS View using Single Parameter

## 1. Introduction

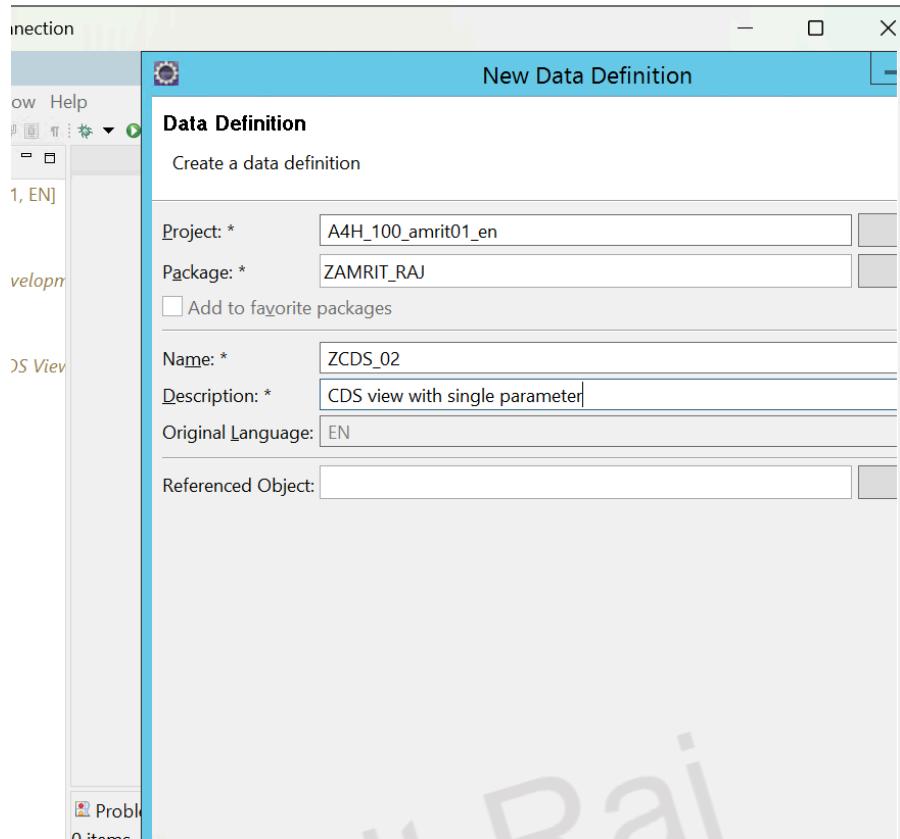
- We can pass parameters to CDS view to take input from the user.
- **\$parameters** keyword is used to refer to the parameter name.

## 2. Requirement

- Display the Sales Order data from VBAK table based on the input given by the user.

### Implementation

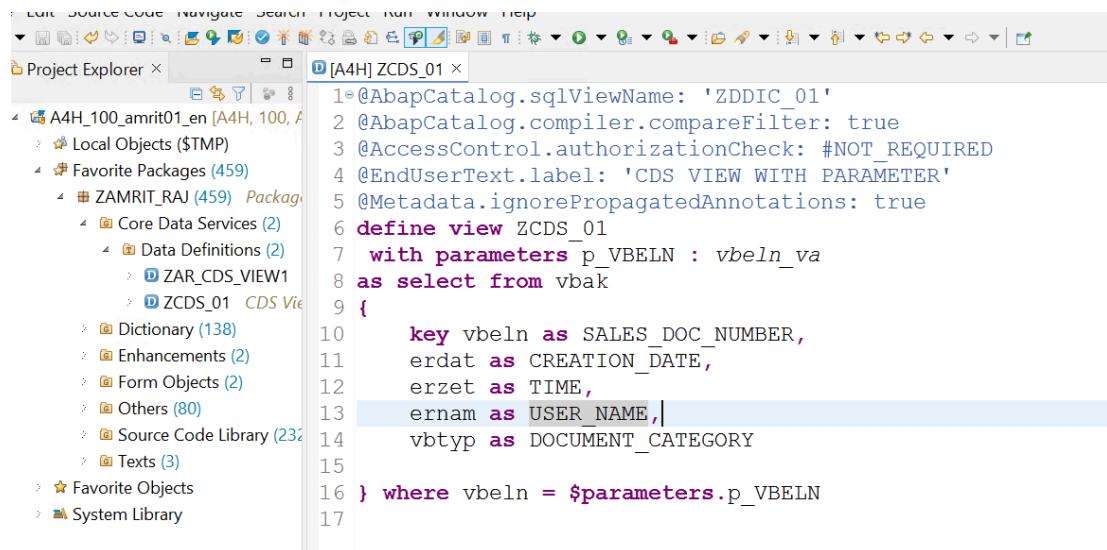
- Step 1 :- Right Click on the Package → New → Data Definition.
  - Pass the name and the description.



- Step 2 :- Click on next and assign the transport request —> Click on next and select Define view and write the logic.

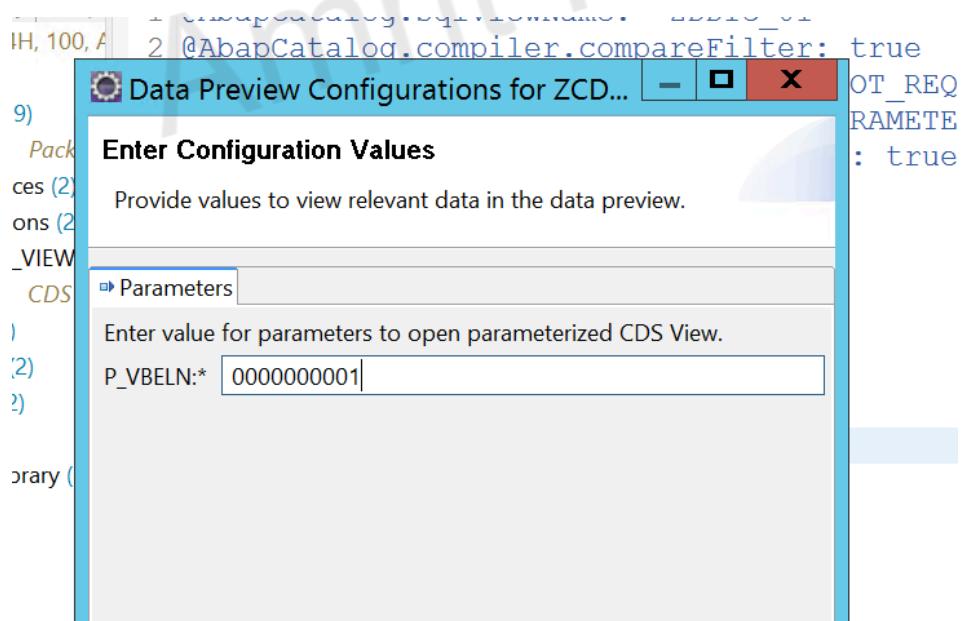
```
@AbapCatalog.sqlViewName: 'ZDDIC_01'  
@AbapCatalog.compiler.compareFilter: true  
@AccessControl.authorizationCheck: #NOT_REQUIRED  
@EndUserText.label: 'CDS VIEW WITH PARAMETER'  
@Metadata.ignorePropagatedAnnotations: true  
define view ZCDS_01  
with parameters p_VBELN : vbeln_va  
as select from vbak  
{  
    key vbeln as SALES_DOC_NUMBER,  
    erdat as CREATION_DATE,  
    erzet as TIME,  
    ernam as USER_NAME,  
    vbtyp as DOCUMENT_CATEGORY
```

```
} where vbeln = $parameters.p_VBELN
```



```
1 @AbapCatalog.sqlViewName: 'ZDDIC_01'
2 @AbapCatalog.compiler.compareFilter: true
3 @AccessControl.authorizationCheck: #NOT_REQUIRED
4 @EndUserText.label: 'CDS VIEW WITH PARAMETER'
5 @Metadata.ignorePropagatedAnnotations: true
6 define view ZCDS_01
7 with parameters p_VBELN : vbeln_val
8 as select from vbak
9 {
10     key vbeln as SALES_DOC_NUMBER,
11     erdat as CREATION_DATE,
12     erzet as TIME,
13     ernam as USER_NAME,
14     vbtyp as DOCUMENT_CATEGORY
15
16 } where vbeln = $parameters.p_VBELN
17
```

- Activate it
- Right click on the view → Open with → Data Preview and pass the parameters.



- Click on Open Data Preview.

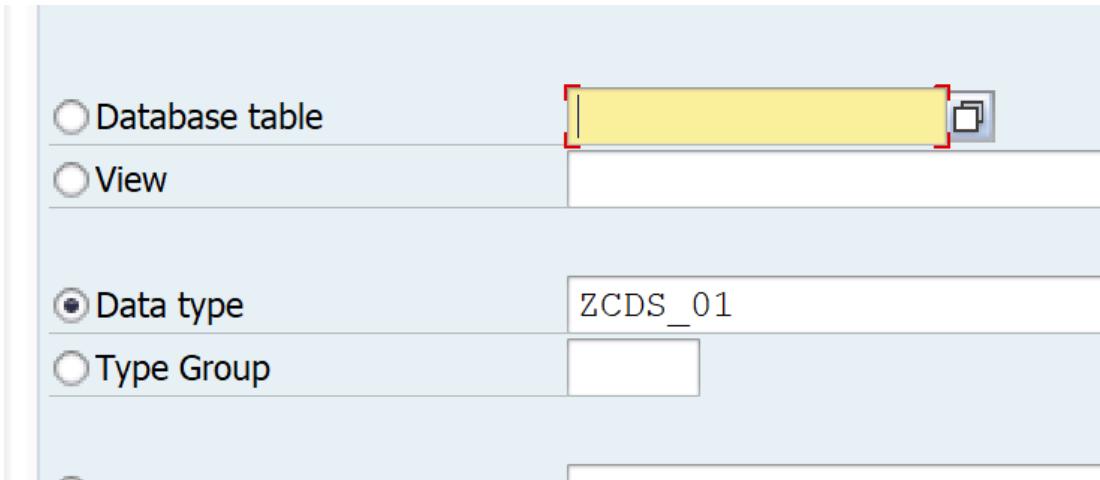
| SALES_DOC_NUMBER | CREATION_DATE | TIME        | USER_NAME | DOCUMENT_CATEGORY |
|------------------|---------------|-------------|-----------|-------------------|
| 0000000001       | 2024-11-20    | 04:22:49... | ABAP10    | C                 |

## Corresponding Data Dictionary View

| View field        | Table | Field |
|-------------------|-------|-------|
| MANDT             | VBAK  | MANDT |
| SALES_DOC_NUMBER  | VBAK  | VBELN |
| CREATION_DATE     | VBAK  | ERDAT |
| TIME              | VBAK  | ERZET |
| USER_NAME         | VBAK  | ERNAM |
| DOCUMENT_CATEGORY | VBAK  | VBTYP |

## How to check the code of CDS View in GUI ?

- Select the data type radio button and pass the name of the cds view



- Open it and go for the content tabs.

**Display Data Definition**

ADT-Link: adt://A4H/sap/bc/adt/ddic/ddl/sources/zcds\_01/source

**Data Definition ZCDS\_01 [A4H, 100]**

```

1 @AbapCatalog.sqlViewName: 'ZDDIC_01'
2 @AbapCatalog.compiler.compareFilter: true
3 @AccessControl.authorizationCheck: #NOT_REQUIRED
4 @EndUserText.label: 'CDS VIEW WITH PARAMETER'
5 @Metadata.ignorePropagatedAnnotations: true
6 define view ZCDS_01
7 with parameters p_VBELN : vbeln_val
8 as select from vbak
9 {
10   key vbeln as SALES_DOC_NUMBER,
11   erdat as CREATION_DATE,
12   erzett as TIME,
13   ernam as USER_NAME,
14   vbttyp as DOCUMENT_CATEGORY
15
16 } where vbeln = $parameters.p_VBELN

```

Use ABAP Development Tools (ADT) in Eclipse to change Data Definition ZCDS\_01



5

## Consume CDS View with Single parameters in ABAP Program

- We can pass parameter value given by the user in the Select Query.

```
PARAMETERS : P_VBELN TYPE VBELN_VA.
```

```
*&Writing Select Query based on CDS Views  
SELECT * FROM ZAR_CDS_VIEW2( P_VBELN = @P_VBELN )  
INTO TABLE @DATA(LT_SALES).
```

```
*&Displaying our data using CL_SALV_TABLE Class  
DATA : LO_TABLE TYPE REF TO CL_SALV_TABLE.
```

```
TRY.  
CALL METHOD CL_SALV_TABLE->FACTORY  
EXPORTING  
LIST_DISPLAY = IF_SALV_C_BOOL_SAP->FALSE
```

```

*   R_CONTAINER    =
*   CONTAINER_NAME =
IMPORTING
  R_SALV_TABLE = LO_TABLE
CHANGING
  T_TABLE      = LT_SALES
.
CATCH CX_SALV_MSG.
ENDTRY.

LO_TABLE->DISPLAY( ).
```

```

8
9  PARAMETERS : P_VBELN TYPE VBELN_VA.
10
11  *&Writing Select Query based on CDS Views
12  SELECT * FROM ZAR_CDS_VIEW2( P_VBELN = @P_VBELN )
13    INTO TABLE @DATA(LT_SALES).
14
15
16  *&Displaying our data using CL_SALV_TABLE Class
17  DATA : LO_TABLE TYPE REF TO CL_SALV_TABLE.
18
19  TRY.
20    CALL METHOD CL_SALV_TABLE=>FACTORY
21      EXPORTING
22        LIST_DISPLAY = IF_SALV_C_BOOL_SAP=>FALSE
23    *     R_CONTAINER    =
24    *     CONTAINER_NAME =
25    IMPORTING
26      R_SALV_TABLE = LO_TABLE
27      CHANGING
28        T_TABLE      = LT_SALES
29      .
30    CATCH CX_SALV_MSG.
31  ENDTRY.

32
33  LO_TABLE->DISPLAY( ).
```

## Output

## ***consuming CDS view in ABAP programming***



P\_VBELN

1

- Click on execute button.

## ***consuming CDS view in ABAP programming***

| Sales Doc. | Created On | Time     | Created By | Doc... |
|------------|------------|----------|------------|--------|
| 1          | 20.11.2024 | 04:22:49 | ABAP10     | C      |



# Aggregate Functions in CDS Views

## 1. Introduction

- Aggregate Functions are mathematical operations that calculates a single value from a group of rows within a database table summarizing data like sums, averages, minimums, maximums, or counts.
- In CDS Views it is mandatory to use **GROUP BY** clause to group data before performing the calculations.

**Various Aggregate Functions are :-**

1. SUM()
2. AVERAGE()
3. MIN()
4. MAX()
5. COUNT()

---

## Requirement

- Use Flight Schedule table { SPFLI } and create a CDS view and use the aggregate functions inside it.

## Without Aggregate Function

```
@AbapCatalog.sqlViewName: 'ZDDIC_01'  
@AbapCatalog.compiler.compareFilter: true  
@AccessControl.authorizationCheck: #NOT_REQUIRED  
@EndUserText.label: 'CDS VIEW WITH PARAMETER'  
@Metadata.ignorePropagatedAnnotations: true  
define view ZCDS_01 as select from spfli  
{  
    key carrid as AIRLINE_CODE,  
    countryfr as COUNTRY_KEY,  
    fltime as FLIGHT_TIME  
}
```

## Output

| AIRLINE_CODE | COUNTRY_KEY | FLIGHT_TIME |
|--------------|-------------|-------------|
| AA           |             | 21,660      |
| AA           |             | 19,260      |
| AC           | DE          | 28,800      |
| AF           | DE          | 39,600      |
| DL           |             | 22,920      |
| DL           |             | 19,500      |
| LH           |             | 30,240      |
| LH           |             | 30,900      |
| LH           |             | 44,400      |
| LH           |             | 48,600      |
| LH           |             | 3,900       |

## With Aggregate Function Sum

```

@AbapCatalog.sqlViewName: 'ZDDIC_01'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'CDS VIEW WITH PARAMETER'
@Metadata.ignorePropagatedAnnotations: true
define view ZCDS_01 as select from spfli
{
    key carrid as AIRLINE_CODE,
    countryfr as COUNTRY_KEY,
    sum(flttime) as FLIGHT_TIME
}

} group by carrid, countryfr

```

## Output

Data Preview

find pattern 8 rows retrieved - 64 ms

| AIRLINE_CODE | COUNTRY_KEY | FLIGHT_TIME |
|--------------|-------------|-------------|
| AA           |             | 40,920      |
| DL           |             | 42,420      |
| LH           |             | 184,740     |
| SQ           |             | 30,000      |
| UA           |             | 116,460     |
| AC           | DE          | 28,800      |
| AF           | DE          | 39,600      |
| LH           | DE          | 3,300       |

## With Having Clause

```

@AbapCatalog.sqlViewName: 'ZDDIC_01'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@endUserText.label: 'CDS VIEW WITH PARAMETER'
@Metadata.ignorePropagatedAnnotations: true
define view ZCDS_01 as select from spfli
{
    key carrid as AIRLINE_CODE,
    countryfr as COUNTRY_KEY,
    sum(flttime) as FLIGHT_TIME

} group by carrid, countryfr
    having sum(flttime) > 40000

```

## Output

The screenshot shows a SAP Studio interface with three tabs at the top: [A4H] ZAR\_CDS\_VIEW2, [A4H] ZAR\_CDS\_VIEW2, and [A4H] ZCDS. Below the tabs, a breadcrumb navigation shows 'ZCDS\_01'. A 'Data Preview' button is highlighted. Below it, a message says '4 rows retrieved - 42 ms'. The main area is a table with three columns: AIRLINE\_CODE, COUNTRY\_KEY, and FLIGHT\_TIME. The data is as follows:

| AIRLINE_CODE | COUNTRY_KEY | FLIGHT_TIME |
|--------------|-------------|-------------|
| AA           |             | 40,920      |
| DL           |             | 42,420      |
| LH           |             | 184,740     |
| UA           |             | 116,460     |

## Using Max, Min, Avg, count

```

@AbapCatalog.sqlViewName: 'ZDDIC_01'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'CDS VIEW WITH PARAMETER'
@Metadata.ignorePropagatedAnnotations: true
define view ZCDS_01 as select from spfli
{
  key carrid as AIRLINE_CODE,
  countryfr as COUNTRY_KEY,
  sum(flttime) as FLIGHT_TIME,
  max(flttime) as MAXIMUM_TIME,
  min(flttime) as MINIMUM_TIME,
  count( distinct flttime) as NUMBER_OF_FLIGHT,
  avg(flttime) as AVERAGE_FLIGHT_TIME

} group by carrid, countryfr
      having sum(flttime) > 40000

```

## Output



| AIRLINE_CODE | COUNTRY_KEY | FLIGHT_TIME | MAXIMUM_TIME | MINIMUM_TIME | NUMBER_OF_FLIGHT | AVERAGE_FLIGHT_TIME   |
|--------------|-------------|-------------|--------------|--------------|------------------|-----------------------|
| LH           |             | 184,740     | 48,600       | 3,900        | 6                | 1.67945454545456E+04  |
| AA           |             | 40,920      | 21,660       | 19,260       | 2                | 2.046000000000000E+04 |
| DL           |             | 42,420      | 22,920       | 19,500       | 2                | 2.121000000000000E+04 |
| UA           |             | 116,460     | 48,900       | 22,200       | 3                | 3.882000000000000E+04 |



7

# Currency Conversion Function using CDS Views

## 1. Introduction

- Currency Conversion is used to convert the currency value from one currency to another currency.
- We have **CURRENCY\_CONVERSION** function which we can use for our purpose.
- For currency conversion we will need
  1. Source currency
  2. Target currency
  3. Exchange rate date
  4. amount
- Currency Exchange rates are maintained in TCURR table.

| Dictionary: Display Table |                                     |                                     |              |           |                  |       |                      |
|---------------------------|-------------------------------------|-------------------------------------|--------------|-----------|------------------|-------|----------------------|
| Transparent Table         |                                     | TCURR                               |              | Active    |                  |       |                      |
| Short Description         |                                     | Exchange Rates                      |              |           |                  |       |                      |
| Attributes                |                                     | Delivery and Maintenance            |              | Fields    | Input Help/Check |       | Currency/Quantity Fi |
|                           |                                     |                                     |              |           |                  |       |                      |
| Field                     | Key                                 | Init...                             | Data element | Data Type | Length           | Decim |                      |
| MANDT                     | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | MANDT        | CLNT      | 3                |       |                      |
| KURST                     | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | KURST CURRE  | CHAR      | 4                |       |                      |
| FCURR                     | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | FCURR CURRE  | CUKY      | 5                |       |                      |
| TCURR                     | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | TCURR CURRE  | CUKY      | 5                |       |                      |
| GDATU                     | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | GDATU INV    | CHAR      | 8                |       |                      |
| UKURS                     | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | UKURS CURRE  | DEC       | 9                |       |                      |
| PRACM                     | <input type="checkbox"/>            | <input type="checkbox"/>            | PRACM CURRE  | DEC       | 0                |       |                      |

## 2. Requirement

- We will use our SFLIGHT table in which we have Price and currency columns.

## 3. Implementation

- Create a CDS View and write the logic.

```

@AbapCatalog.sqlViewName: 'ZDDIC_VIEW_01'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'First CDS View to display Materia'
@Metadata.ignorePropagatedAnnotations: true
define view ZAR_CDS_VIEW1
with parameters P_CURR : abap.cuky,
      P_DATE : abap.dats
as select from sflight
{
  key sflight.carrid,

```

```

key sflight.connid,
key sflight.fldate,
@Semantics.amount.currencyCode: 'TARGET_CURRENCY'
currency_conversion( AMOUNT => sflight.price,
                     SOURCE_CURRENCY => sflight.currency,
                     TARGET_CURRENCY => :P_CURR,
                     EXCHANGE_RATE_DATE => :P_DATE ) as CONV_CURR,
@Semantics.currencyCode: true
:P_CURR as TARGET_CURRENCY

} where carrid = 'AC'

```

## Output

The screenshot shows the SAP Studio interface with two main panes. On the left, a code editor displays the CDS View definition:

```

'A4H] ZAR_CDS_VIEW3 [A4H] ZAR_CDS_VIEW1 ×
4 @AccessControl.authorizationCheck: #NOT_REQUIRED
5 @EndUserText.label: 'First CDS View to display Material...
6 @Metadata.ignorePropagatedAn...
7 define view ZAR_CDS_VIEW1
8 with parameters P_CURR : ab...
9          P_DATE : ab...
0 as select from sflight
1 {
2     key sflight.carrid,
3     key sflight.connid,
4     key sflight.fldate,
5     @Semantics.amount.curren...
6     currency_conversion( AM...
7         SOU...
8         TAR...
9         EXC...
0     @Semantics.currencyCode: ...
1     :P_CURR as TARGET_CURREN...
2
3

```

On the right, a modal dialog titled "Data Preview Configurations for ZAR..." is open. It has a title bar with a close button and a "Data Preview" button. The dialog is divided into sections:

- Enter Configuration Values**: A note: "Provide values to view relevant data in the data preview."
- Parameters**: A section for entering parameter values.
- P\_CURR**: Set to "USD".
- P\_DATE**: Set to "20250205".
- Buttons**: Includes a question mark icon, "Open Data Preview" (highlighted in blue), and a "Cancel" button.

- Click on Open data preview.

 Data Preview

 find pattern  1 rows retrieved - 112 ms

| RB | carrid | RB   | connid | FL | fldate      | 12 | CONV_CURR | RB  | TARGET_CURRENCY |
|----|--------|------|--------|----|-------------|----|-----------|-----|-----------------|
| AC |        | 0820 |        |    | 2002-12-... |    | 816.84    | USD |                 |
|    |        |      |        |    |             |    |           |     |                 |
|    |        |      |        |    |             |    |           |     |                 |
|    |        |      |        |    |             |    |           |     |                 |

Amrit Raj



8

## Extend CDS View

- When we have to add more columns into the table, then we go for adding append structure to extend our table.
- Similarly, we can also extend our CDS view by adding more columns.

## Implementation

- Step 1 :- Create a new data definition.
- Step 2 :- Select the template extend CDS View.

Use the selected template

| Name                                   | Description                                   |
|--|---|
| defineRootViewEntity                   | Define Root View Entity                       |
| defineViewEntityWithToParentAssociatic | Define View Entity with To-Parent Association |
| defineView                             | Define View (obsolete as of AS ABAP 7.5)      |
| >  Projection View (creation)          |   |
| ◀  View Extend (creation)              |   |
| extendViewEntity                       | Extend View Entity                            |
| extendView                             | Extend View                                   |
| >  Table Function (creation)           |   |

```

@AbapCatalog.sqlViewAppendName: '${sql_view_append}'
@EndUserText.label: '${ddl_source_description}'
extend view ${view_name} with ${ddl_source_name_edited}
{
    ${base_data_source_name}.${element_name}
}

```

- Click on Finish button.
- Step 3 :- Add the columns.

```

Source Code Navigate Search Project Run Window Help
[44H] ZAR_CD... [44H] ZAR_CD... [44H] ZAR_CD... [44H] ZAR_CD... [44H] ZAR_CD...
1 @AbapCatalog.sqlViewAppendName: 'ZDDIC_01_EX'
2 @EndUserText.label: 'CDS View'
3 extend view ZAR_CDS_VIEW2 with ZAR_CDS_VIEW5
4 {
5     augru,
6     vkorg
7 }
8

```

The screenshot shows the SAP Studio interface with the code editor open. The code editor displays a SQL script for extending a CDS view. The script includes the `@AbapCatalog.sqlViewAppendName` and `@EndUserText.label` annotations, followed by the `extend view` statement and its associated code block. The code block contains two column names: `augru` and `vkorg`. The SAP Studio interface includes a menu bar, toolbars, and a sidebar showing project navigation.

## Output

Data Preview

find pattern 1 rows retrieved - 27 ms

| SALES_DOC_NUMBER | CREATION_DATE | TIME        | USER_NAME | D. | augru | vkorg |
|------------------|---------------|-------------|-----------|----|-------|-------|
| 0000000001       | 2024-11-20    | 04:22:49... | ABAP10    | C  |       | 1710  |

## Note

- If you will check your DDIC view, you will find our that columns has been appended into the view.

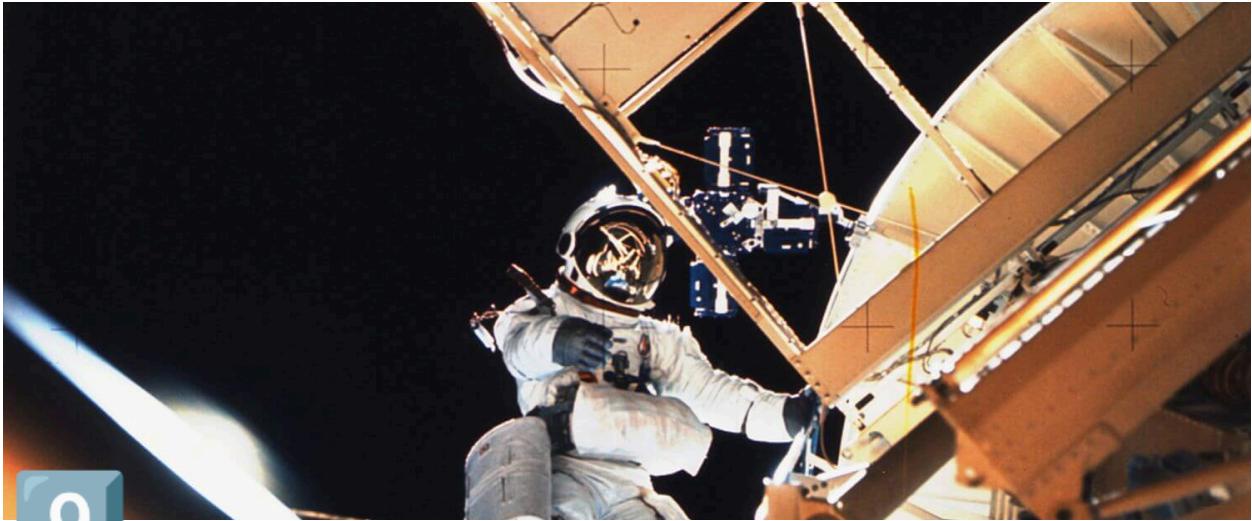
CDS Database View ZDDIC\_01\_EX Active

Short Description CDS View

Attributes View Fields

Tables

| View field        | Table Name  | Field Name |
|-------------------|-------------|------------|
| TIME              | VBAK        | ERZET      |
| USER_NAME         | VBAK        | ERNAM      |
| DOCUMENT_CATEGORY | VBAK        | VBTyp      |
| .APPEND           | ZDDIC_01_EX | *          |
| AUGRU             | VBAK        | AUGRU      |
| VKORG             | VBAK        | VKORG      |



9

## CDS Views with Joins

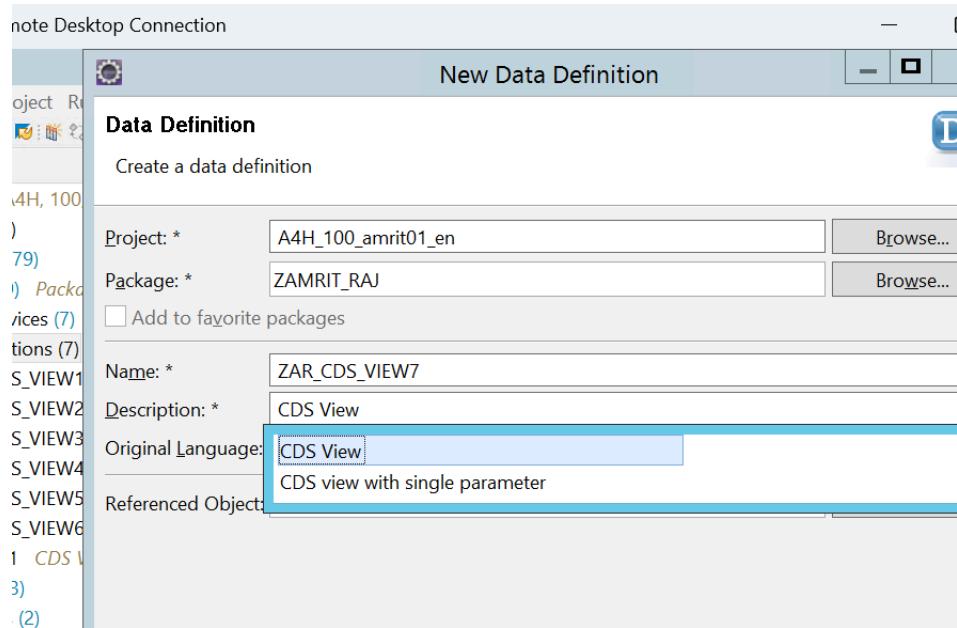
- In ABAP Programming Joins, are used to fetch data from multiple tables using Single Select Query based on a common condition between the tables.
- Similarly, we can also use Joins in CDS Views to fetch data from Multiple tables.

### Requirement

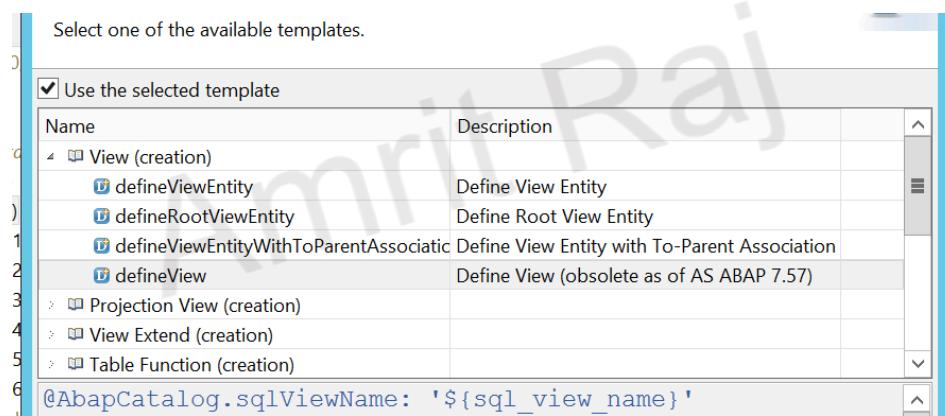
- Display the Employee and Project Details data using CDS View.

### Implementation

- Step 1 :- Create a new data definition.



- Step 2 :- Assign the transport request and select the define view.



- Step 3 :- Write the logic.

```

@AbapCatalog.sqlViewName: 'ZDDIC_01'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'CDS VIEW'
@Metadata.ignorePropagatedAnnotations: true
define view ZCDS_01
with parameters P_EMP_ID : zar_emp_id
as select from zar_employee_tab
inner join zar_project_det
    
```

```

on zar_employee_tab.emp_id = zar_project_det.emp_id
{
    zar_employee_tab.emp_id,
    zar_employee_tab.emp_name,
    zar_employee_tab.department,
    zar_employee_tab.manager,
    zar_employee_tab.salary,
    zar_employee_tab.currency,
    zar_project_det.project_id,
    zar_project_det.project_name
} where zar_employee_tab.emp_id = $parameters.P_EMP_ID

```

## Output

| emp\_id | emp\_name | department | manager | salary | currency | project\_id | project\_name |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 101 | SHIVAM SINGH | SAP ABAP | AMRIT RAJ | 10000.00 | USD | 100 | NESTLE |
| 101 | SHIVAM SINGH | SAP ABAP | AMRIT RAJ | 10000.00 | USD | 200 | HITACHI |
| 101 | SHIVAM SINGH | SAP ABAP | AMRIT RAJ | 10000.00 | USD | 300 | COCA-COLA |

## Note

- Similarly, We can write the logic for Left outer join and Right outer also.



10

# Joins on more than 2 tables using CDS Views and consuming it in ABAP Report

- In the previous session, we have implemented joins on two tables Employee and Project Details.
- In this Part we will implemented Joins on three standard tables { VBAK, VBAP, MAK } and we will fetch out the data based on the input given by user for Sales Document Number.

## Implementation

- Step 1 :- Create a New Data Definition.
- Step 2 :- Write the logic based on the input given by the user.

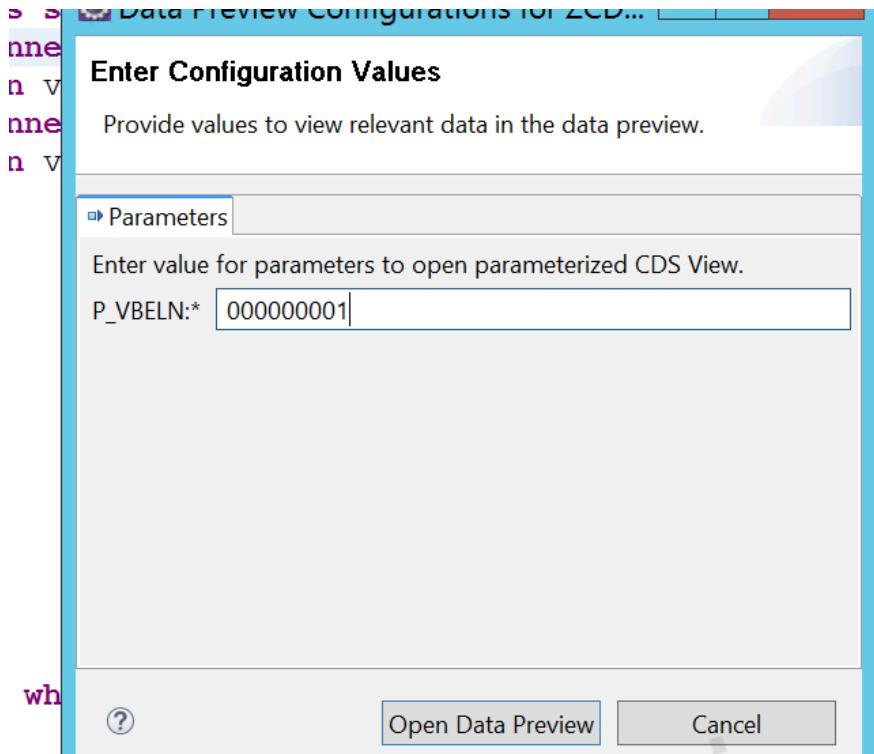
```
@AbapCatalog.sqlViewName: 'ZDDIC_01'  
@AbapCatalog.compiler.compareFilter: true  
@AccessControl.authorizationCheck: #NOT_REQUIRED  
@EndUserText.label: 'CDS VIEW'  
@Metadata.ignorePropagatedAnnotations: true
```

```
define view ZCDS_01
with parameters P_VBELN : vbeln_va
as select from vbak
inner join vbap
on vbak.vbeln = vbap.vbeln
inner join makt
on vbap.matnr = makt.matnr
{
    vbak.vbeln as SALES_DOC_NUM,
    vbak.erdat as CREATION_DATE,
    vbak.erzet as TIME,
    vbak.ernam as USER_NAME,
    vbak.vbtyp as DOCUMENT_CATEGORY,

    vbap.posnr as ITEM_NUMBER,
    vbap.matnr as MATERIAL_NUMBER,

    makt.spras as LANGUAGE_KEY,
    makt.maktx as MATERIAL_DESCRIPTION
} where vbak.vbeln = $parameters.P_VBELN
```

## Output



- Click on Open Data Preview.

| SALES_DOC_NUM | CREATION_DATE | TIME        | USER_NAME | DO... | ITEM_NUMBER | MATERIAL_NUMBER   | LANGUAGE_KEY | MATERIAL_DESCRPT            |
|---------------|---------------|-------------|-----------|-------|-------------|-------------------|--------------|-----------------------------|
| 0000000001    | 2024-11-20    | 04:22:49... | ABAP10    | C     | 000010      | AVC_RBT_BRD_PAINT | E            | Control Board (Painting)    |
| 0000000001    | 2024-11-20    | 04:22:49... | ABAP10    | C     | 000010      | AVC_RBT_BRD_PAINT | D            | Bedienungspult (Lackierung) |

## Corresponding DDIC View Generated

- Go to SE11 and Open the DDIC View.

| View field        | Table | Field | Key                                 | Data elem. |
|-------------------|-------|-------|-------------------------------------|------------|
| MANDT             | VBAK  | MANDT | <input checked="" type="checkbox"/> | MANDT      |
| SALES_DOC_NUM     | VBAK  | VBELN | <input type="checkbox"/>            | VBELN_VA   |
| CREATION_DATE     | VBAK  | ERDAT | <input type="checkbox"/>            | ERDAT      |
| TIME              | VBAK  | ERZET | <input type="checkbox"/>            | ERZET      |
| USER_NAME         | VBAK  | ERNAM | <input type="checkbox"/>            | ERNAM      |
| DOCUMENT_CATEGORY | VBAK  | VBTYP | <input type="checkbox"/>            | VBTYPL     |

## Consuming our CDS View in ABAP Program

\*&Taking input from the user

```
PARAMETERS : P_VBELN TYPE VBELN_VA.
```

\*&Writing Select Query by using new syntax

```
SELECT *
FROM ZCDS_01( P_VBELN = @P_VBELN )
INTO TABLE @DATA(LT_SALES).
```

\*&Display the data to the user

```
CL_DEMO_OUTPUT→DISPLAY( LT_SALES ).
```

## Output

|                |   |
|----------------|---|
| Sales Document | 1 |
|----------------|---|

- Click on execute button

Output

| LT_SALES      |               |          |           |                   |             |  |
|---------------|---------------|----------|-----------|-------------------|-------------|--|
| SALES_DOC_NUM | CREATION_DATE | TIME     | USER_NAME | DOCUMENT_CATEGORY | ITEM_NUMBER | MATERIAL_NUMBER LANGUAGE_KEY MATERIAL_DESCRIPTION  |
| 0000000001    | 2024-11-20    | 04:22:49 | ABAP10    | C                 | 000010      | AVC_RBT_BRD_PAINT E<br>Control Board (Painting)    |
| 0000000001    | 2024-11-20    | 04:22:49 | ABAP10    | C                 | 000010      | AVC_RBT_BRD_PAINT D<br>Bedienungspult (Lackierung) |

Amrit Raj