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 ne 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.

2️⃣

Creating CDS View

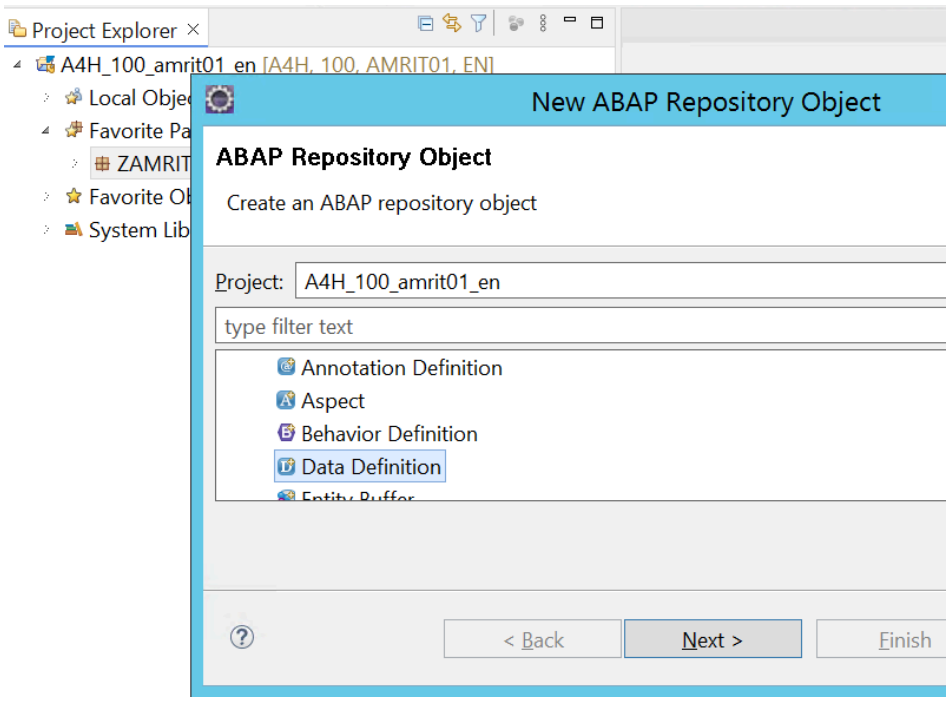
- Whenever we create a CDS View in ADT, A corresponding DDIC View is created into the SAP System.

Requirement

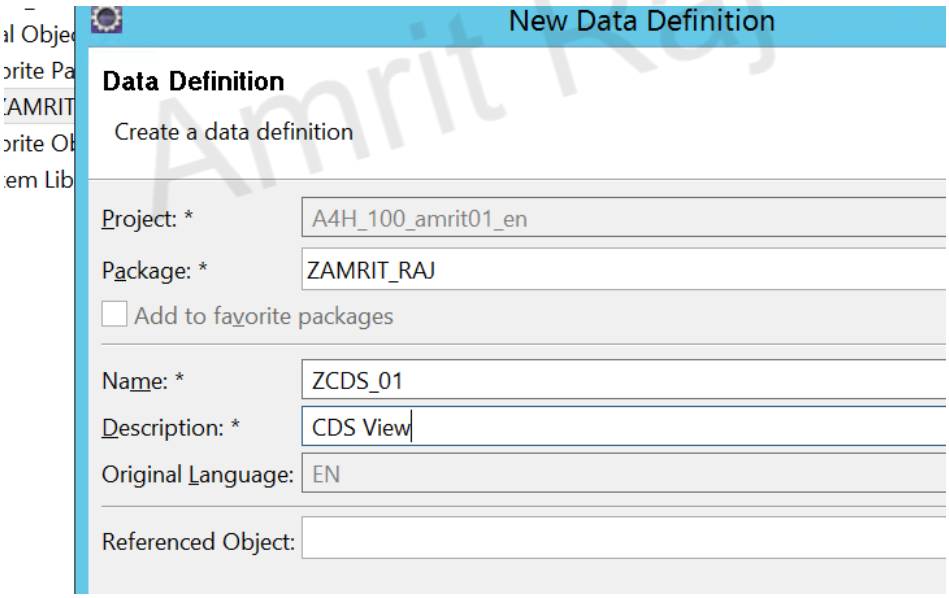
- Display the Material Master data using the CDS View.

Implementation

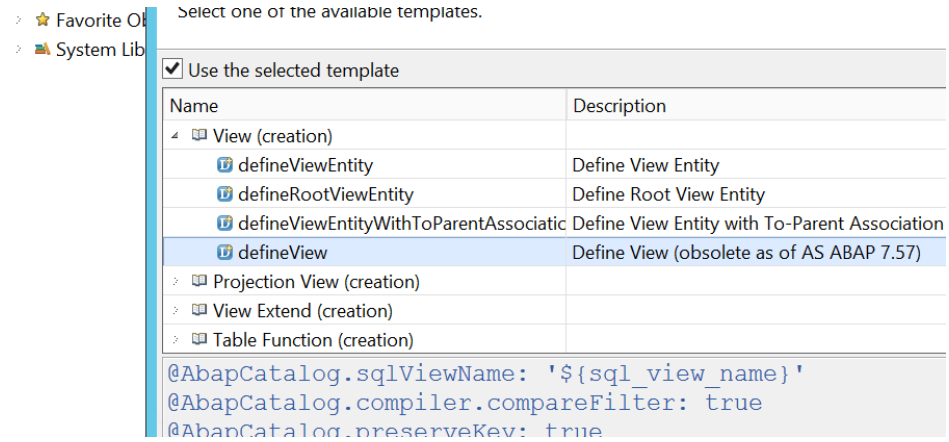
- Step 1 :- Right click on the package → New → Data Definition



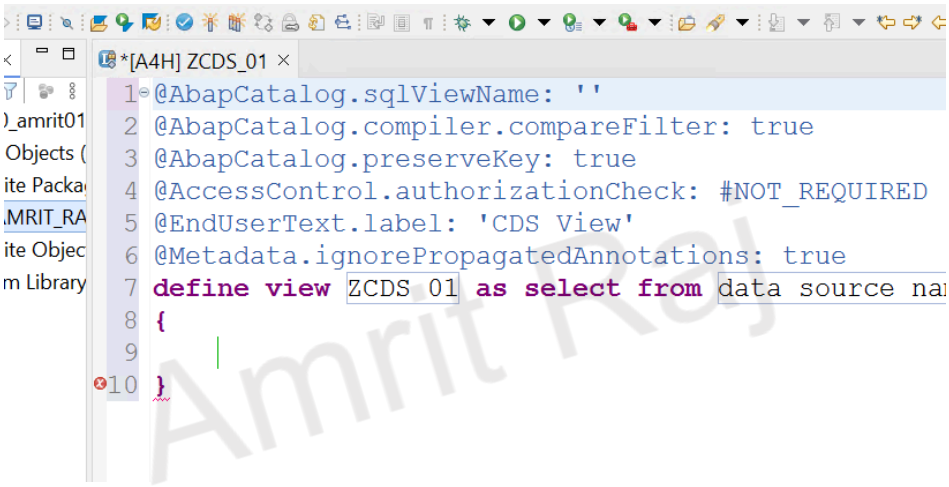
- Step 2 :- Pass the name of the CDS Views and short description.



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



- Click on Finish.



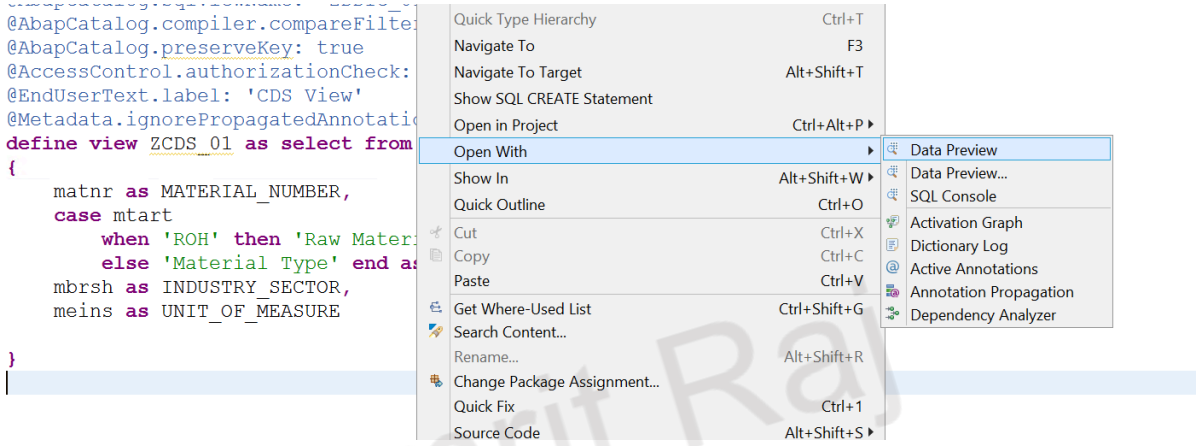
- Step 5 :- Write the logic.

```
@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

}
```

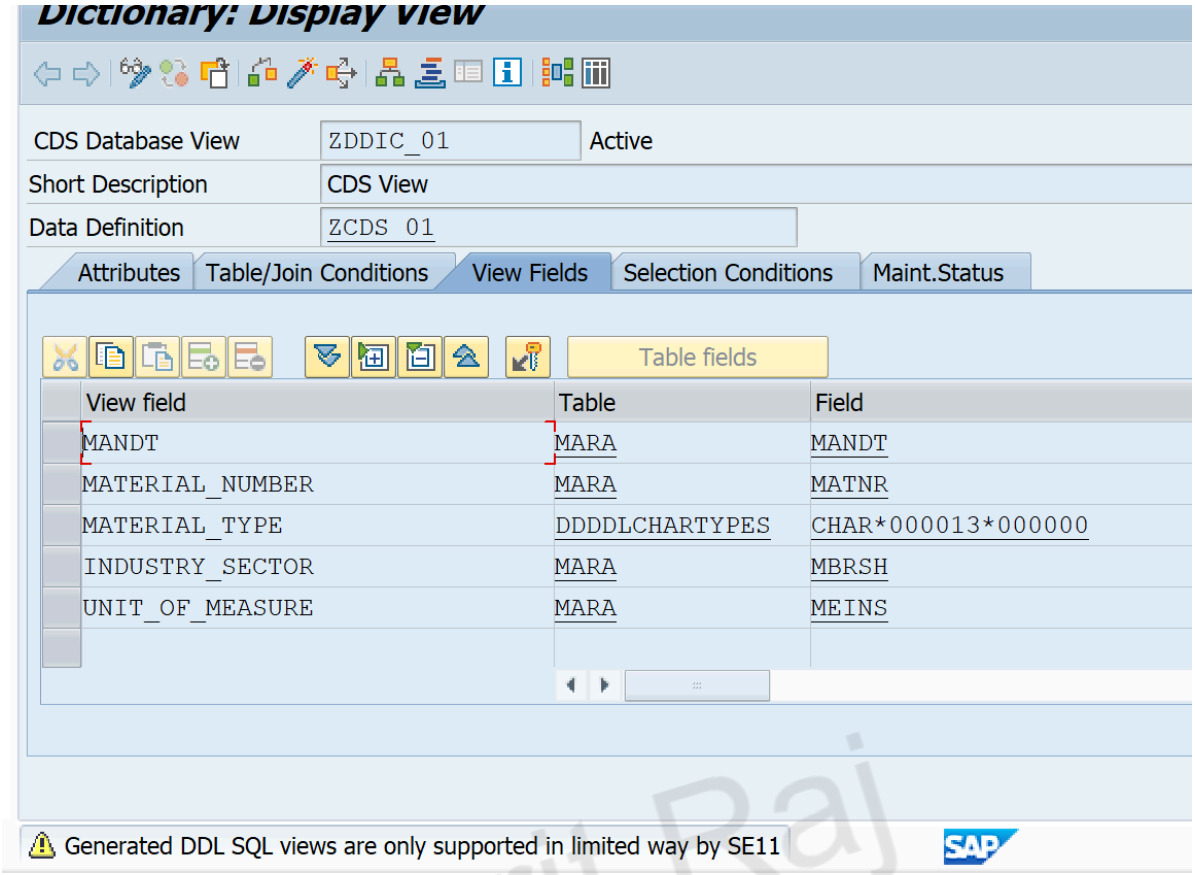
- Activate it.

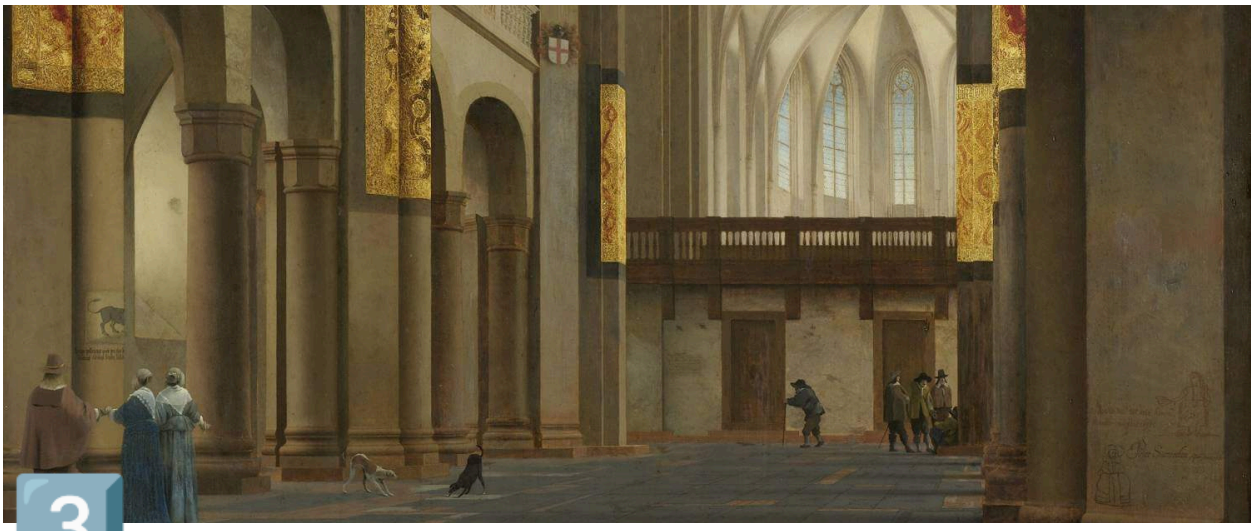
- Step 6 :- Right click → Open with data preview.



Note

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





3️⃣

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.

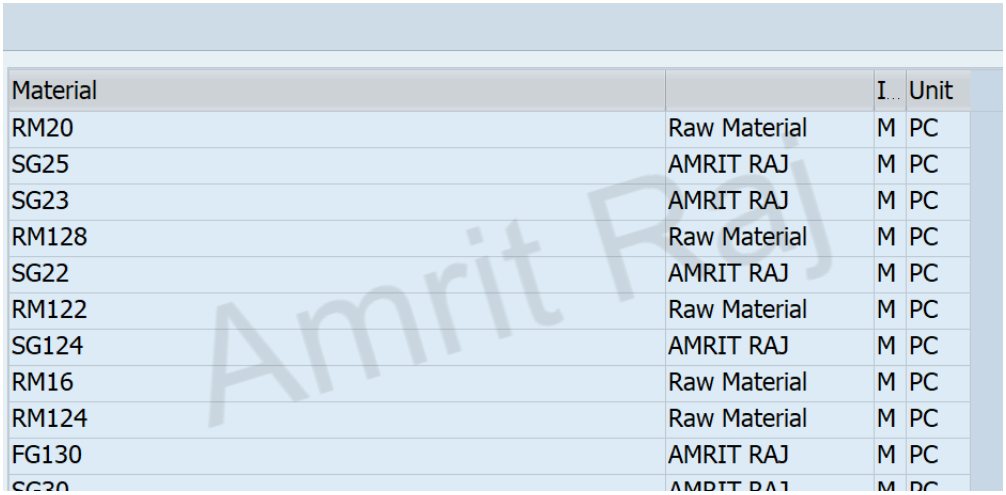
Implementation

- 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 Mo
*      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
SG30	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 to refer to the parameter name.

---

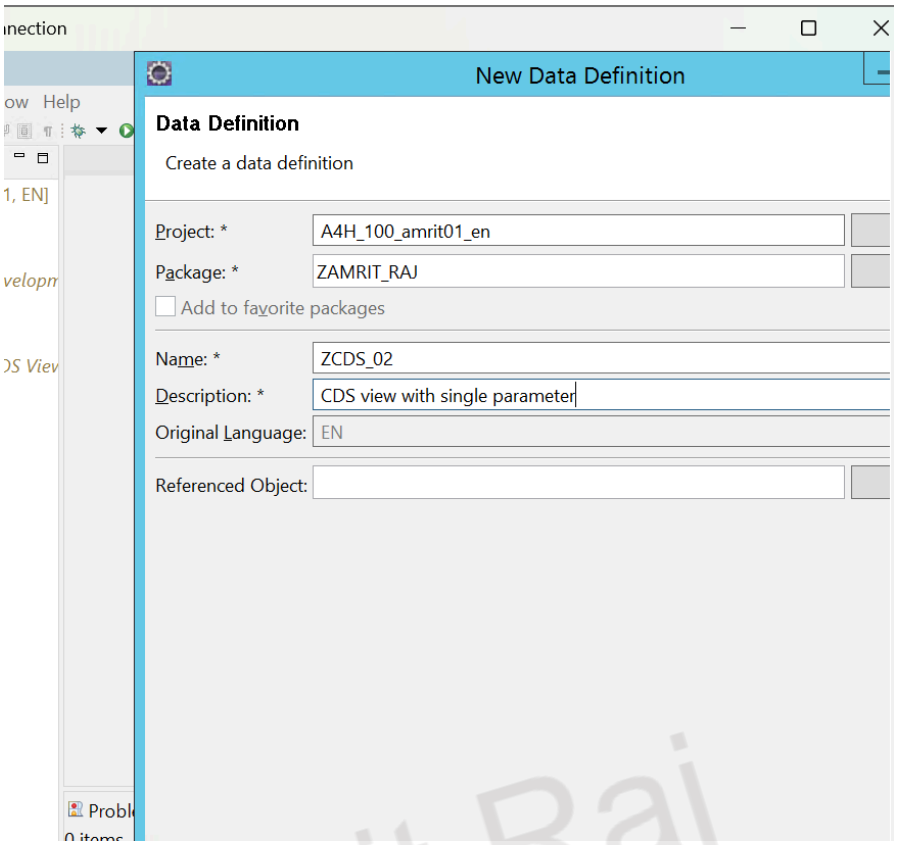
2. Requirement

- 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.

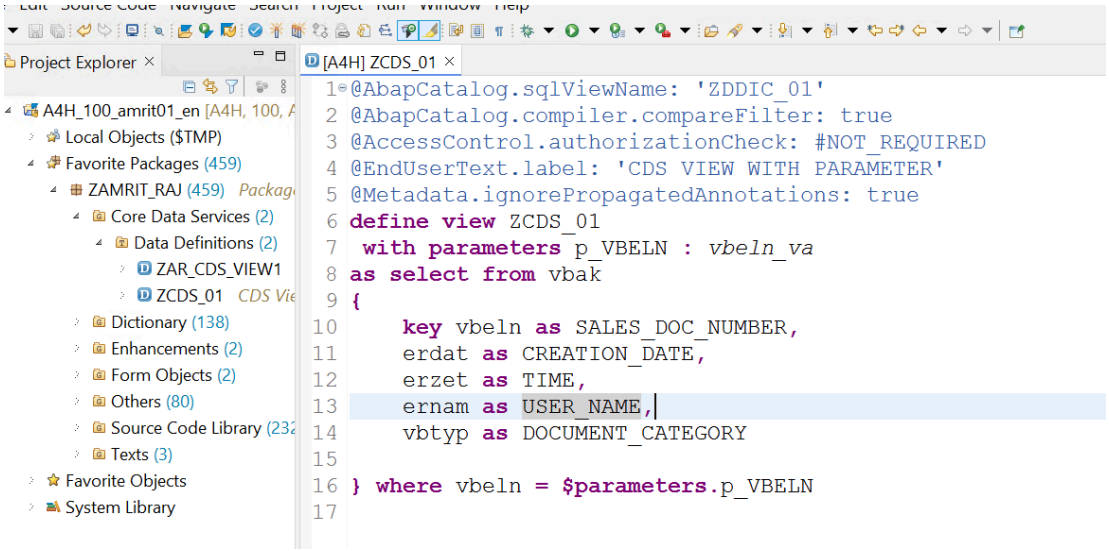
Mob No :- +916261538504



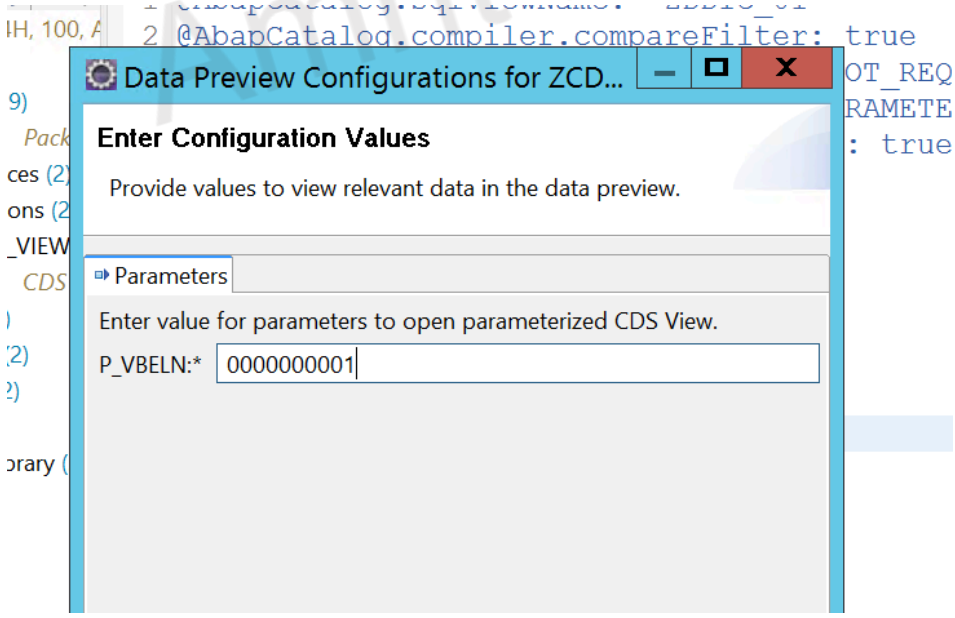
- Step 2 :- Click on next and assign the transport request —> Click on next and select Define view and write the logic.

```
@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
```

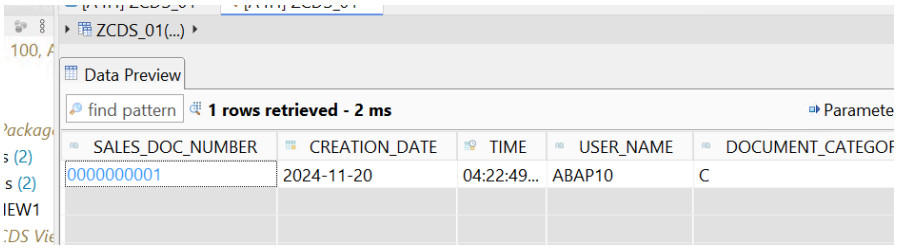
```
} where vbeln = $parameters.p_VBELN
```



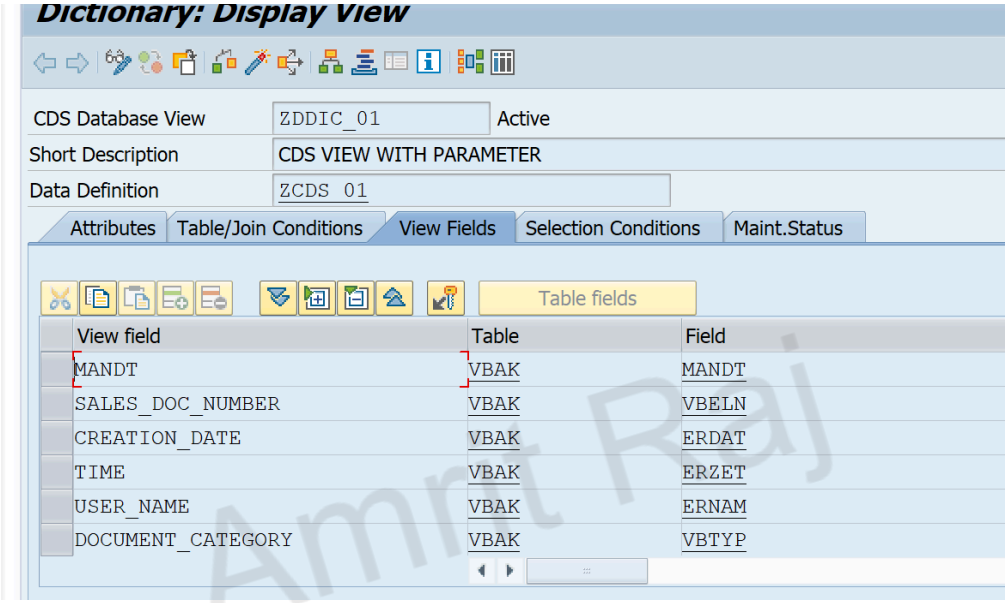
- Activate it
- Right click on the view → Open with → Data Preview and pass the parameters.



  - Click on Open Data Preview.

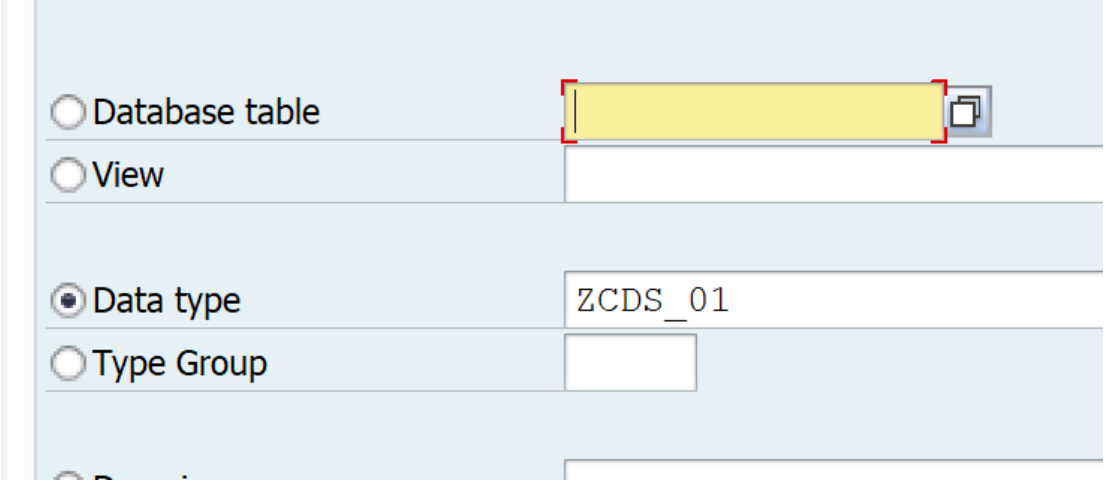


Corresponding Data Dictionary View

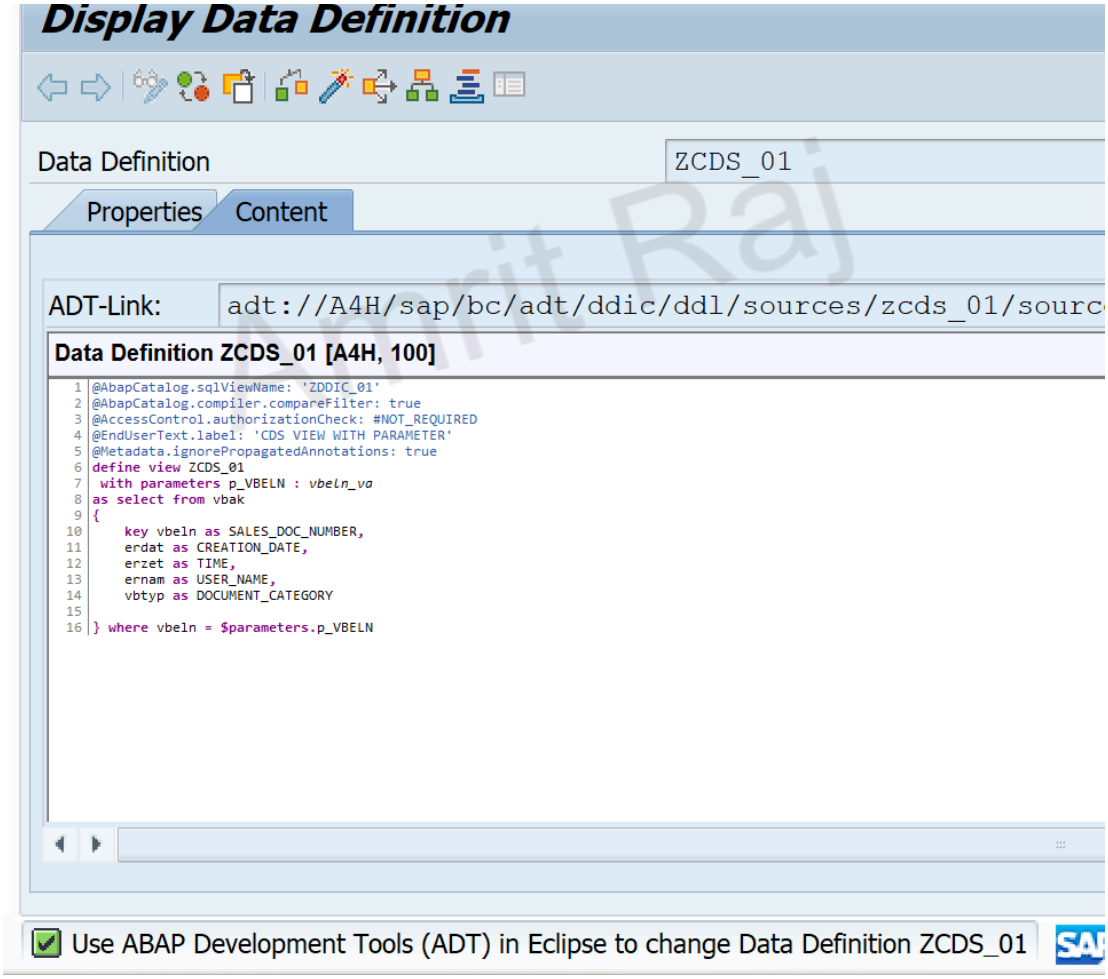


How to check the code of CDS View in GUI ?

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



- Open it and go for the content tabs.



5️⃣

Consume CDS View with Single paramters 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
```

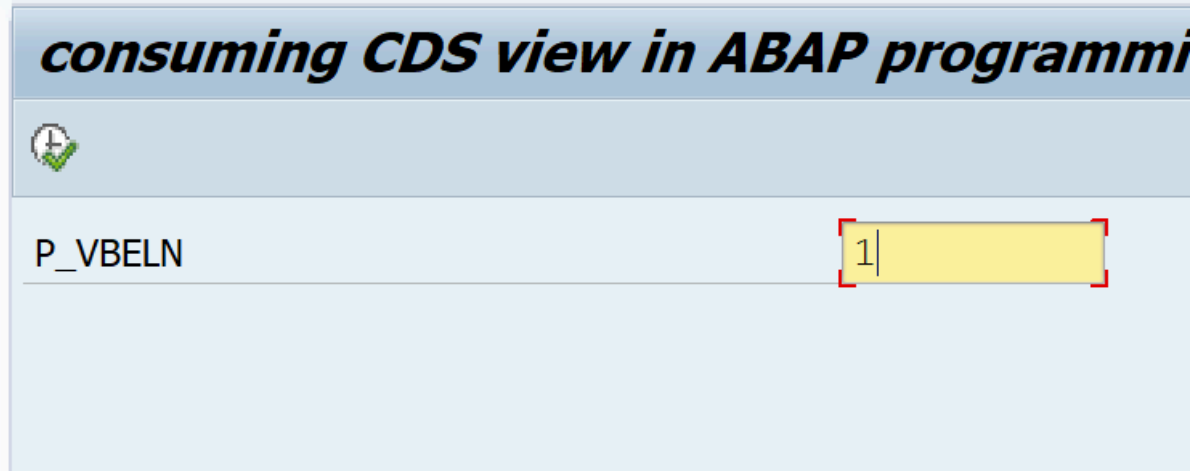

Mob No :- +916261538504

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

  LO_TABLE→DISPLAY( ).
```

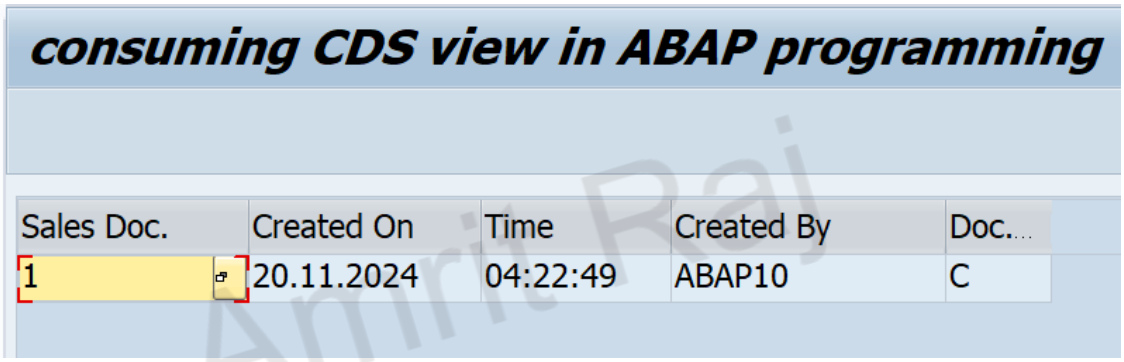
```
 8
 9   PARAMETERS : P_VBELN TYPE VBELN_VA.
10
11   *&Writing Select Query based on CDS Views
12   SELECT * FROM ZAR_CDS_VIEW2( P_VBELN = @P_VBELN )
13     INTO TABLE @DATA(LT_SALES).
14
15
16   *&Displaying our data using CL_SALV_TABLE Class
17     DATA : LO_TABLE TYPE REF TO CL_SALV_TABLE.
18
19     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

Mob No :- +916261538504



- Click on execute button.



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

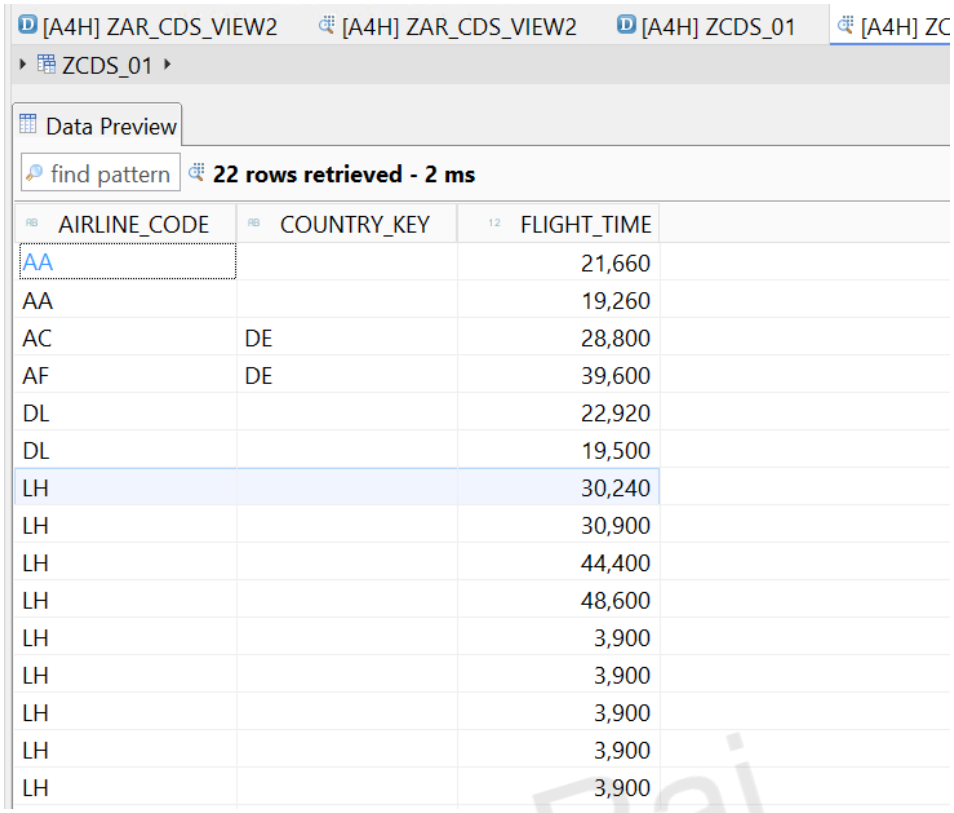
Mob No :- +916261538504

- 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



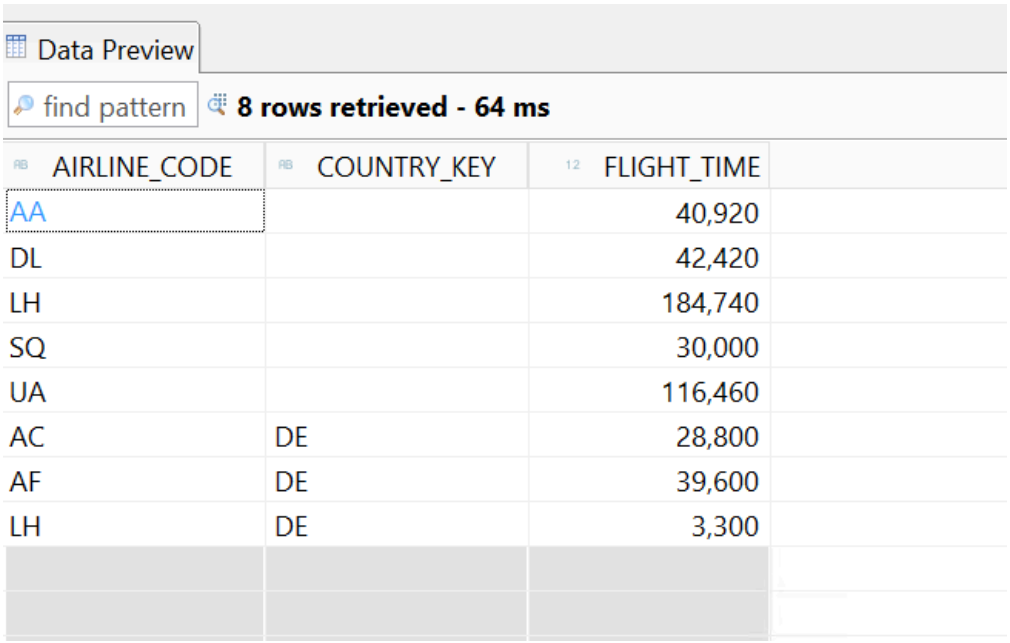
AIRLINE_CODE	COUNTRY_KEY	FLIGHT_TIME
AA		21,660
AA		19,260
AC	DE	28,800
AF	DE	39,600
DL		22,920
DL		19,500
LH		30,240
LH		30,900
LH		44,400
LH		48,600
LH		3,900
LH		3,900
LH		3,900
LH		3,900
LH		3,900

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(fltime) as FLIGHT_TIME


} group by carrid, countryfr
```

Output

Mob No :- +916261538504

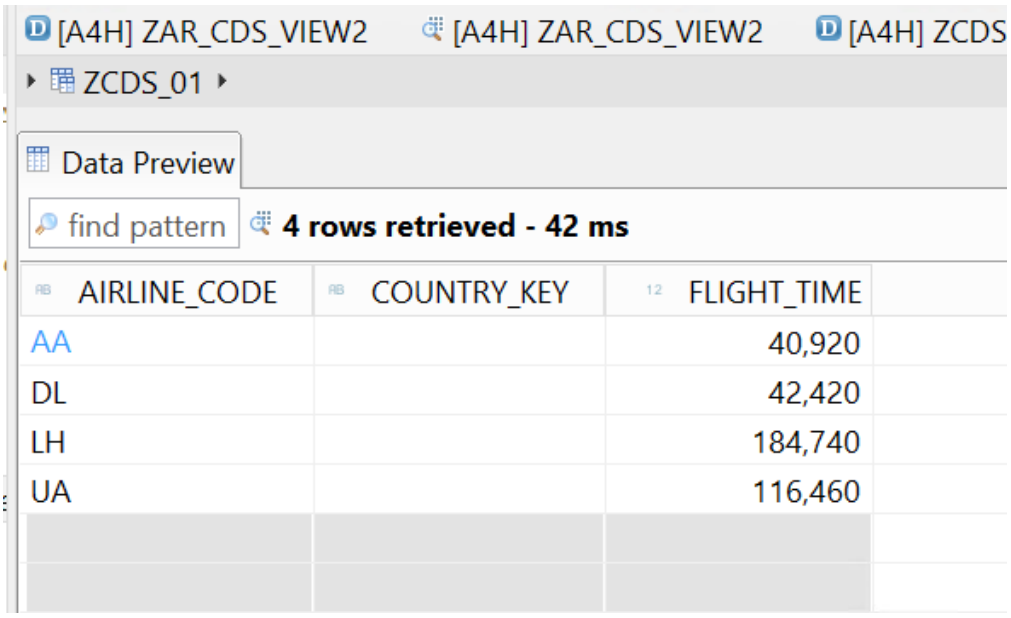


AIRLINE_CODE	COUNTRY_KEY	FLIGHT_TIME
AA		40,920
DL		42,420
LH		184,740
SQ		30,000
UA		116,460
AC	DE	28,800
AF	DE	39,600
LH	DE	3,300

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(fltime) as FLIGHT_TIME


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

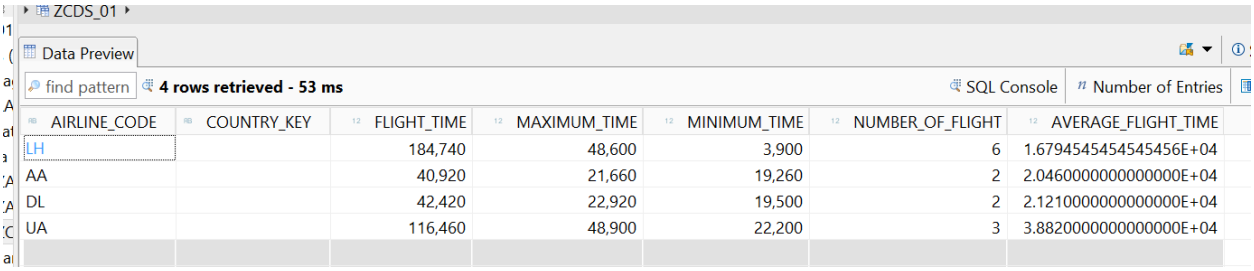
Output



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(fltime) as FLIGHT_TIME,
   max(fltime) as MAXIMUM_TIME,
   min(fltime) as MINIMUM_TIME,
   count( distinct fltime) as NUMBER_OF_FLIGHT,
   avg(fltime) as AVERAGE_FLIGHT_TIME


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

Output


ZCDS_01

Data Preview

find pattern | 4 rows retrieved - 53 ms | SQL Console | Number of Entries

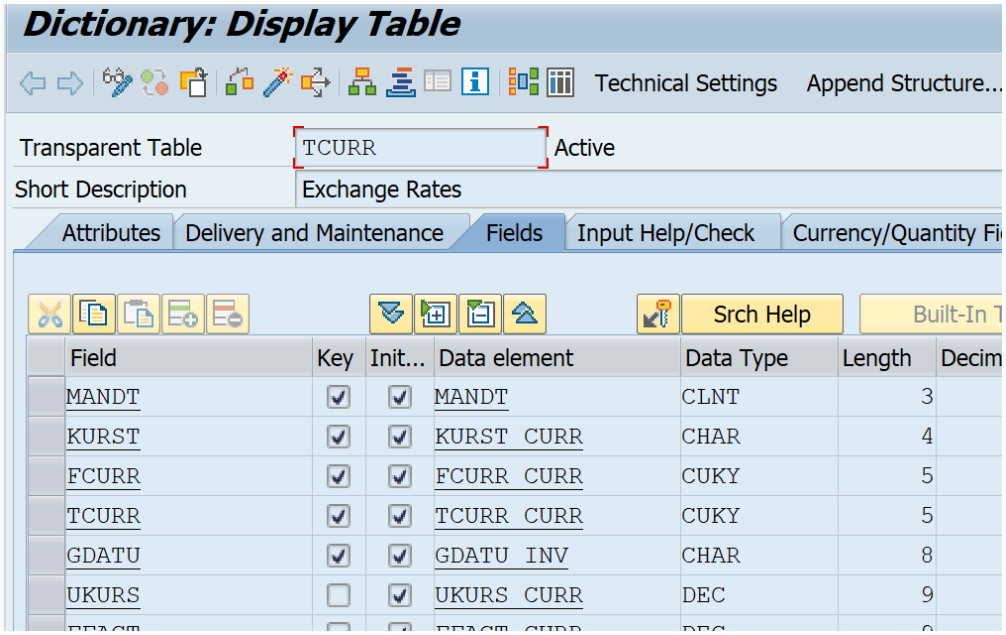
AIRLINE_CODE	COUNTRY_KEY	FLIGHT_TIME	MAXIMUM_TIME	MINIMUM_TIME	NUMBER_OF_FLIGHT	AVERAGE_FLIGHT_TIME
LH		184,740	48,600	3,900	6	1.6794545454545456E+04
AA		40,920	21,660	19,260	2	2.0460000000000000E+04
DL		42,420	22,920	19,500	2	2.1210000000000000E+04
UA		116,460	48,900	22,200	3	3.8820000000000000E+04

7️⃣

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.



2. Requirement

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

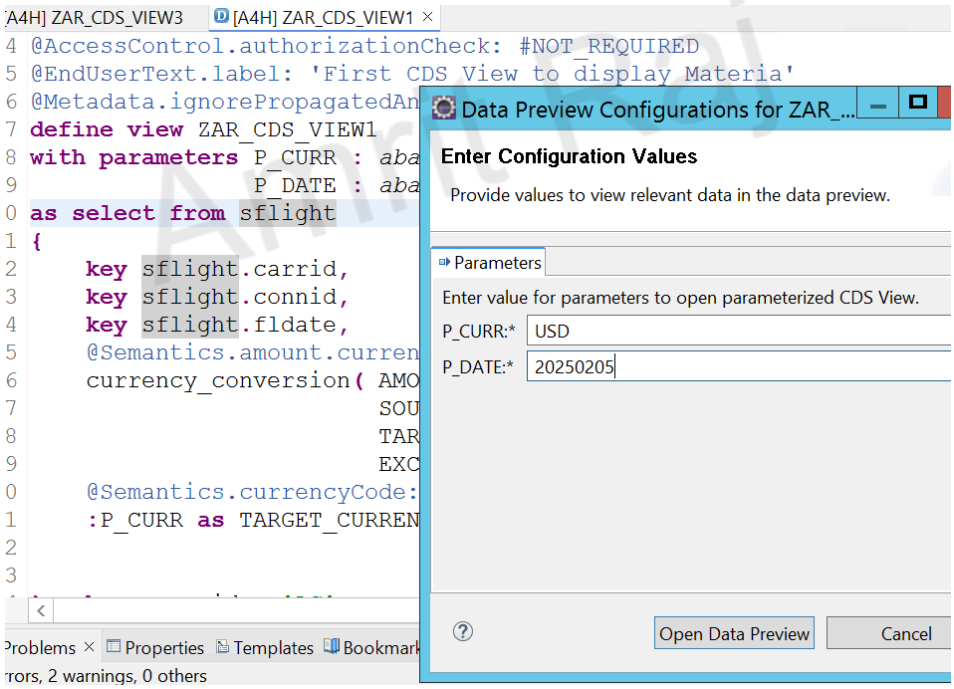
3. Implementation

- Create a CDS View and write the logic.

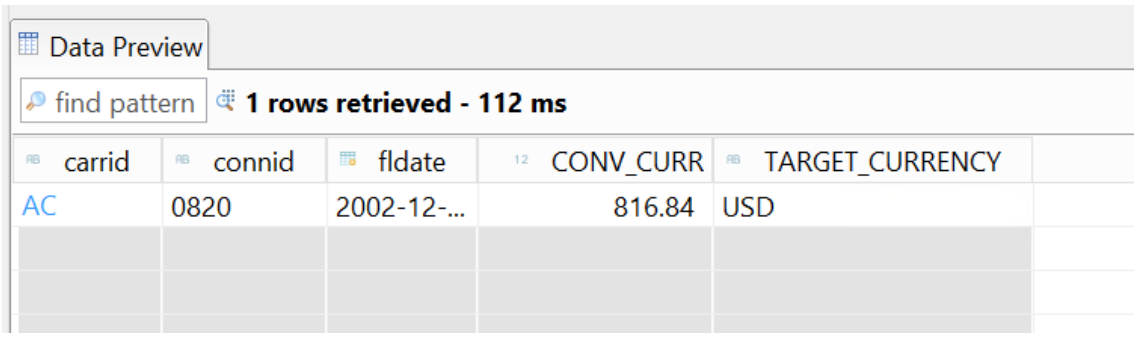
```
@AbapCatalog.sqlViewName: 'ZDDIC_VIEW_01'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'First CDS View to display Materia'
@Metadata.ignorePropagatedAnnotations: true
define view ZAR_CDS_VIEW1
with parameters 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



- Click on Open data preview.



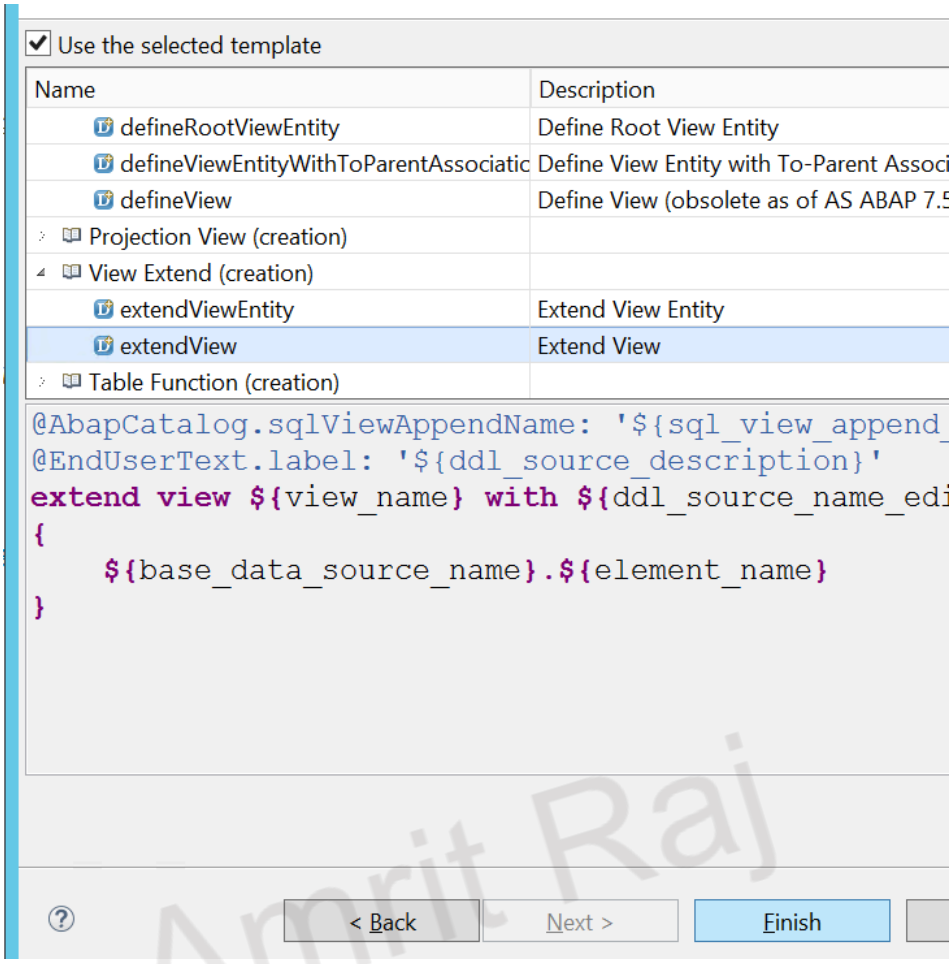
Extend CDS View

- 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.

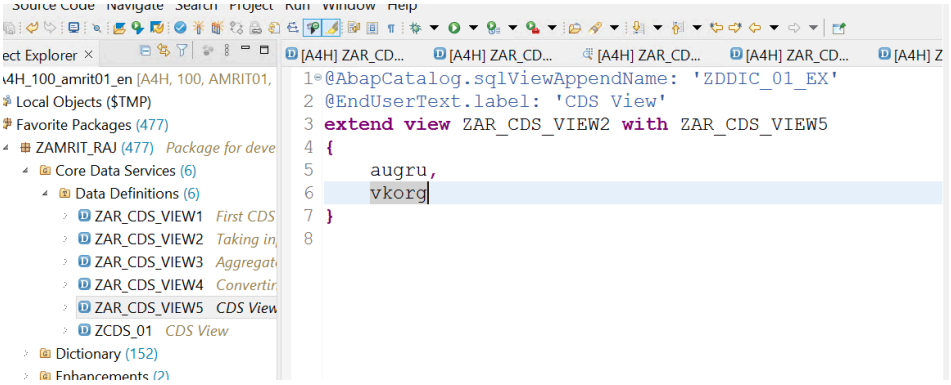
Implementation

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

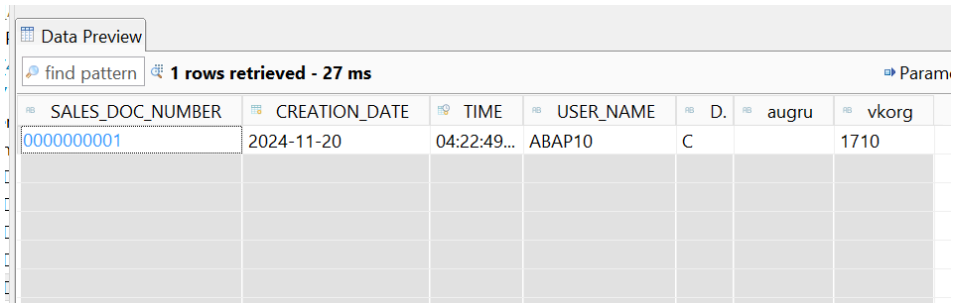
Mob No :- +916261538504



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

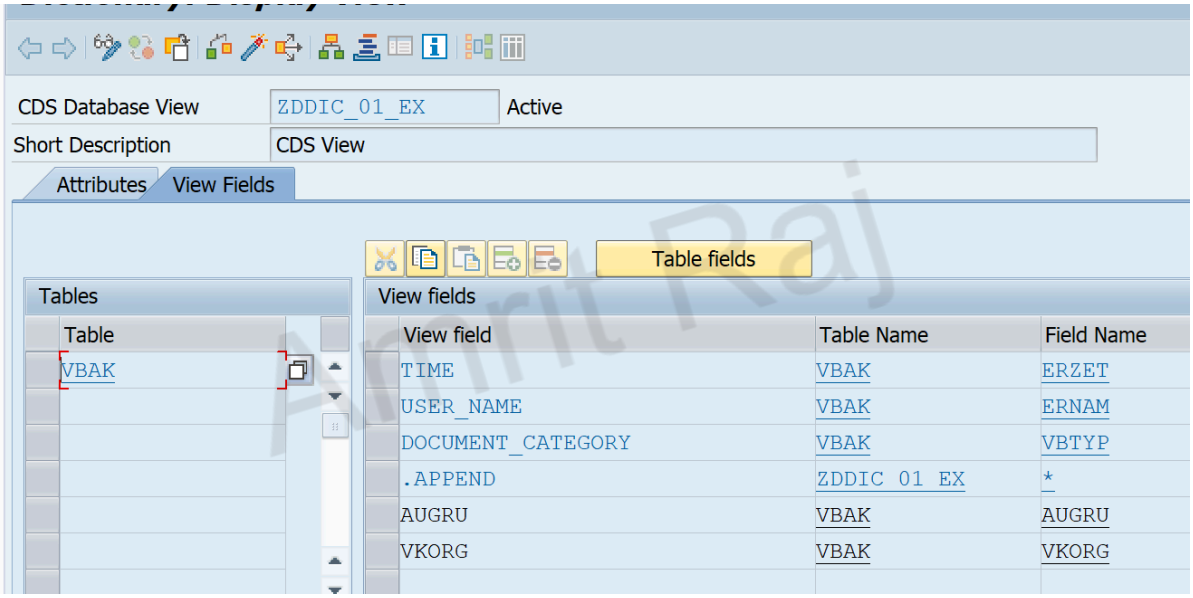


Output



Note

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



9️⃣

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.

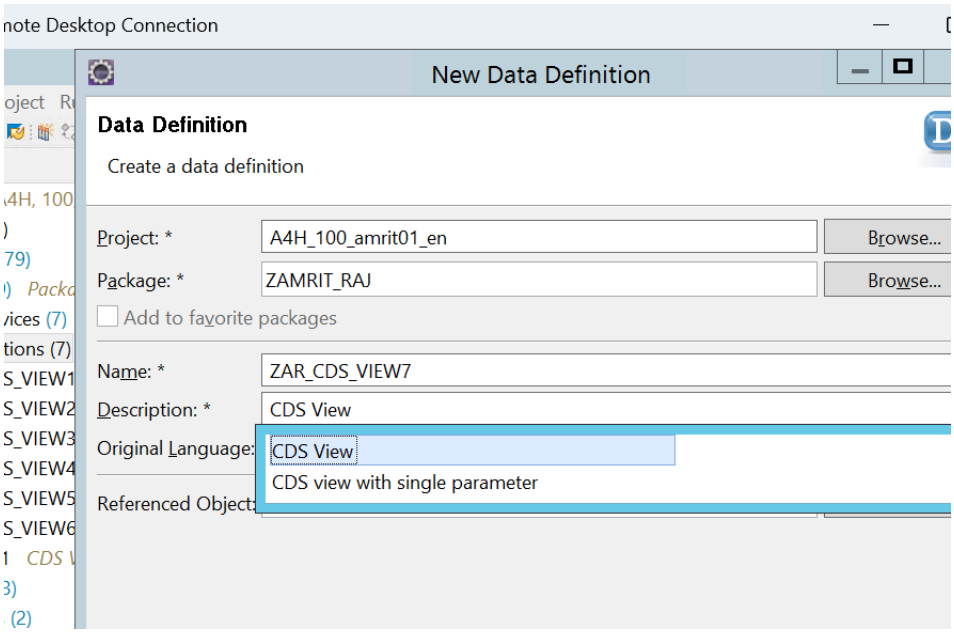
Requirement

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

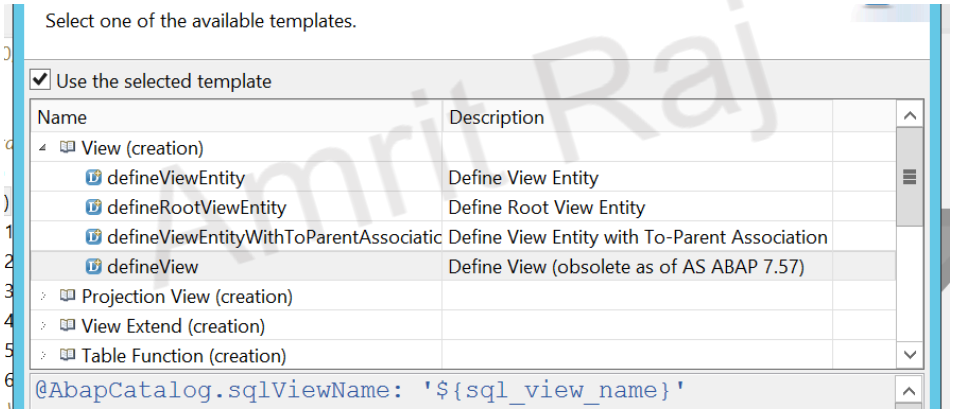
Implementation

- Step 1 :- Create a new data definition.

Mob No :- +916261538504



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

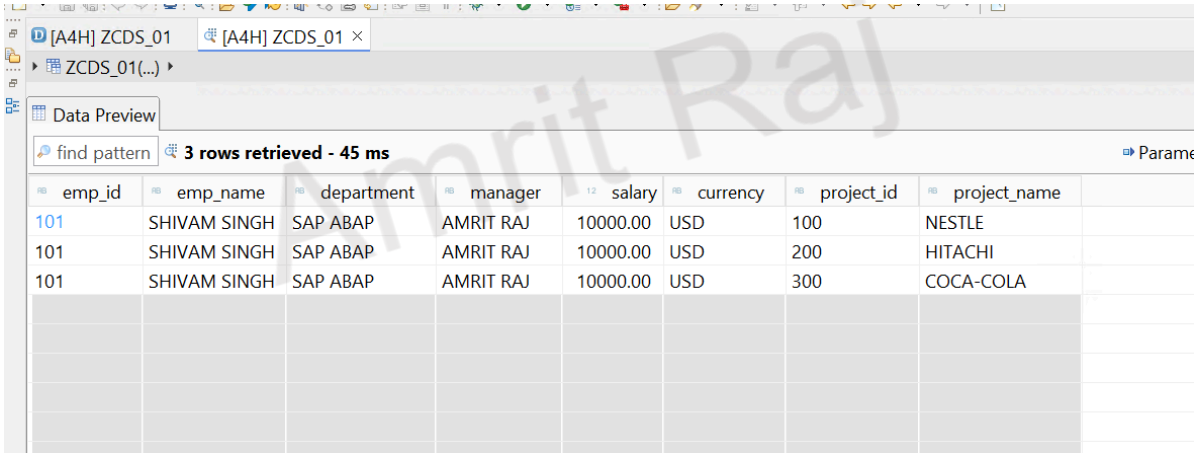


  - Step 3 :- Write the logic.

```
@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.

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, MAKT } 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.

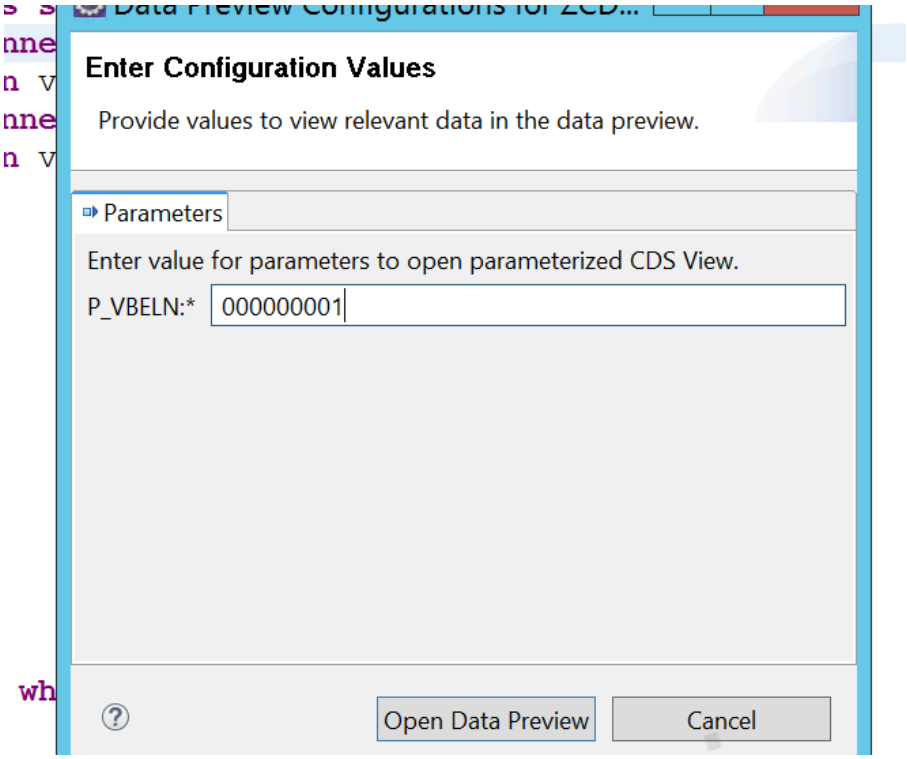
```
@AbapCatalog.sqlViewName: 'ZDDIC_01'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'CDS VIEW'
@Metadata.ignorePropagatedAnnotations: true
```


Mob No :- +916261538504

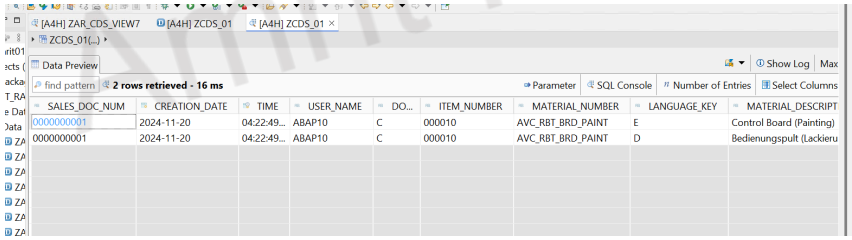
```
define view ZCDS_01
with parameters P_VBELN : vbeln_va
as select from vbak
inner join vbap
on vbak.vbeln = vbap.vbeln
inner join makt
on vbap.matnr = makt.matnr
{
   vbak.vbeln as SALES_DOC_NUM,
   vbak.erdat as CREATION_DATE,
   vbak.erzet as TIME,
   vbak.ernam as USER_NAME,
   vbak.vbtyp as DOCUMENT_CATEGORY,

   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

Mob No :- +916261538504



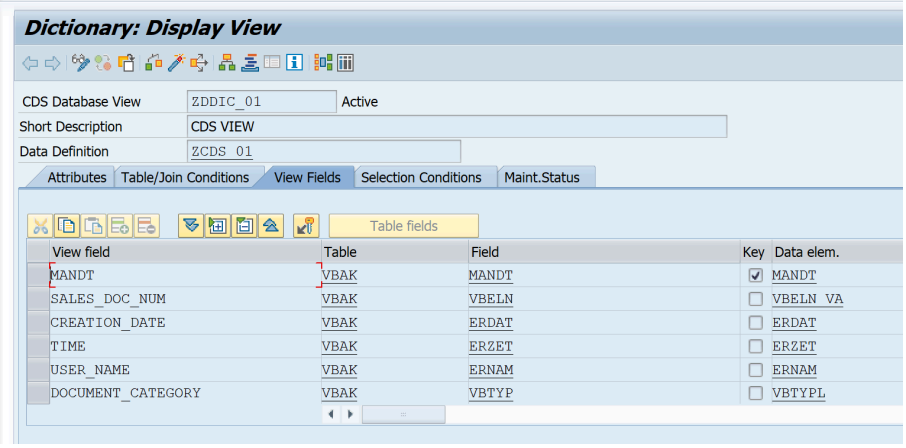
- Click on Open Data Preview.



Corresponding DDIC View Generated

- Go to SE11 and Open the DDIC View.

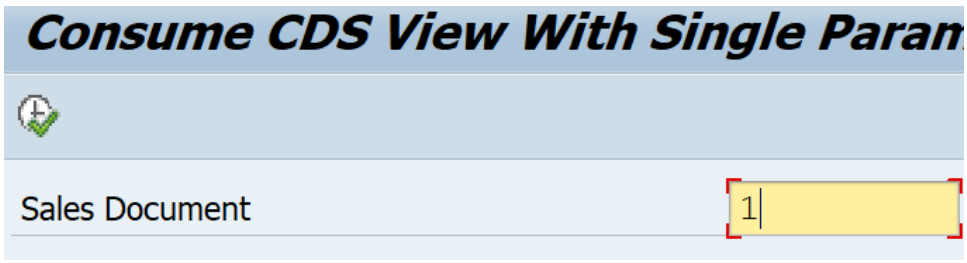
Mob No :- +916261538504



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



- Click on execute button

Mob No :- +916261538504