SQL *LOADER:

SQL Loader *is a tool from Oracle that helps load data into database tables from an external file. To use SQL Loader*, you need two files:

1. **A data file**: This file contains the actual records or data you want to load into the table. (e.g., CSV, TXT).
2. **A control file**: This file tells SQL*Loader how to interpret the data file and where to put the data in the table.(e.g., CTL)

When you run SQL*Loader with the control file, it creates **three different output files** to help track the load process:

1. Log File:

- This file keeps a record of how the data load went. It shows:
  - The number of records read from the data file.
  - The number of records successfully loaded into the table.
  - Any errors or warnings during the load.
- If the load process is interrupted, the log file helps you know where the load stopped, so you can resume from there.

2. Bad File (Reject File):

- This file contains the records that **could not be loaded** due to errors.

3. Discard File:

- This file holds records that **did not meet certain conditions** set in the control file.
- For example, if you have a condition that only loads records for employees in the "HR" department, any records that do not meet this condition will go into the discard file.

Before designing an interface for loading data into a database, here are the **main components** you should consider:

1. **Data File**: The file containing the data to be loaded (e.g., CSV, TXT).
2. **File Format**: The format of the data file (e.g., CSV, tab-delimited, XML).

3. **Fixed or Variable Length**: Whether the data in the file has fixed-length or variable-length fields.
4. **Data Types & Width**: The data types (e.g., VARCHAR, NUMBER) and the size/length of each column in the database.
5. **Frequency**: How often the data is updated or loaded (e.g., hourly, daily, weekly).
6. **File Location**: Where the data file is stored (local, remote server, cloud storage).
7. **Database Table**: The target table in the database where the data will be loaded.

Structure of Data File – Fixed Length

- The data file can be in fixed record format or variable record format.
- Fixed Record Format would look like the below. In this case you give a specific position where the Control file can expect a data field:

```
7369 SMITH     CLERK      7902  12/17/1980     800
7499 ALLEN     SALESMAN   7698  2/20/1981      1600
7521 WARD      SALESMAN   7698  2/22/1981      1250
7566 JONES     MANAGER    7839  4/2/1981       2975
7654 MARTIN    SALESMAN   7698  9/28/1981      1250
```

Structure of Data File – Variable Length(Delimiter Based)

- Variable Record Format would like below where the data fields are separated by a delimiter.
- Note: The Delimiter can be anything you like. In this case it is "|"

```
1196700|9|0|692.64
1378901|2|3900|488.62
1418700|2|2320|467.92
1418702|14|8740|4056.36
1499100|1|0|3.68
```

Structure of Control File:

1. LOAD DATA

- This statement indicates that you are loading data from an external file into the database.

2. INFILE

The **INFILE** keyword is used in the control file to specify the location of the **data file** that SQL*Loader will read from.

i. ***INFILE :*** When you use `INFILE *`, it means that the data is **inside the control file itself** and not in an external file. This is useful when you want to include data directly in the control file rather than referencing an external file.

ii. **INFILE '$FILE':** You can use `INFILE '$FILE'` when the file path and name are **passed as parameters** to the control file, especially when running the SQL*Loader as a **concurrent program** in Oracle EBS or another application.
The `$FILE` is typically a **substitution variable** that gets replaced with the actual file path at runtime.

iii. **INFILE '/home/vision/xxltech/GLData.csv':**
This specifies the **exact file path** and name of the **data file**. SQL*Loader will read the file from the provided location (`/home/vision/xxltech/GLData.csv`).

3. APPEND (or REPLACE, INSERT, TRUNCATE)

- Determines how the data will be loaded into the database table:
  - `APPEND`: Adds new rows to the existing data. If data already exists, it appends the new rows.
  - `REPLACE`: Deletes all existing rows in the table and loads the new data.
  - `INSERT`: Inserts new rows into the table. TABLE MUST BE EMPTY BEFORE INSERTING OTHERWISE INSERT DOES NOT TAKES PLACE.
  - `TRUNCATE`: Removes all data from the table before loading new data.

4. INTO TABLE "APPS"."GL_INTERFACE"

- Specifies the **target table** in the database where data will be loaded.

5. FIELDS TERMINATED BY ‘,’

- Defines the **delimiter** used in the data file. In this case, a comma (`','`), typically for CSV files.

6. OPTIONALLY ENCLOSED BY ‘”’

- If the data fields are enclosed by double quotes (e.g., `"12345"`), this clause handles that.

7. TRAILING NULLCOLS

- For records with missing values, this option treats those missing columns as **NULL**.

8. Column Definitions:

- You can apply SQL functions or transformations to data while loading:
  - For example: `ITEM_NUMBER “TRIM(:ITEM_NUMBER)”` removes leading or trailing spaces from the `ITEM_NUMBER`.
  - You can also use constants, like `DIVISION_CODE CONSTANT "AUD"` to insert a fixed value instead of reading it from the data file.

9. FILLER Keyword:

  - You can skip columns using the ‘FILLER’ KEYWORD. E.G.

```
LOAD DATA
INFILE 'data.csv'
REPLACE
INTO TABLE XXLT_EMP_TBL
FIELDS TERMINATED BY ','
TRAILING NULLCOLS
(
   name      CHAR,     -- This column will be loaded
   Empno     INTEGER,   -- This column will be loaded
   sal       INTEGER,   -- This column will be loaded
   FILLER    CHAR,     -- This column will be skipped (address column)
)
```

Examples of Control File:

Example where datafile is an external file:

LOADDATA statement is required at the beginning of the control file.

INFILE: INFILE keyword is used to specify location of the datafile or datafiles.

```
LOAD DATA

INFILE  '/home/vision/kap/import2.csv'

INTO TABLE kap_emp

FIELDS TERMINATED BY ","

( emp_num, emp_name, department_num, department_name )
```

Example where datafile is in the Control file:

```
LOAD DATA
INFILE *
INTO TABLE kap_emp
FIELDS TERMINATED BY ","
( emp_num, emp_name, deptno , deptname)

BEGINDATA

7369,SMITH,7902,Accounting

7499,ALLEN,7698,Sales

7521,WARD,7698,Accounting
```

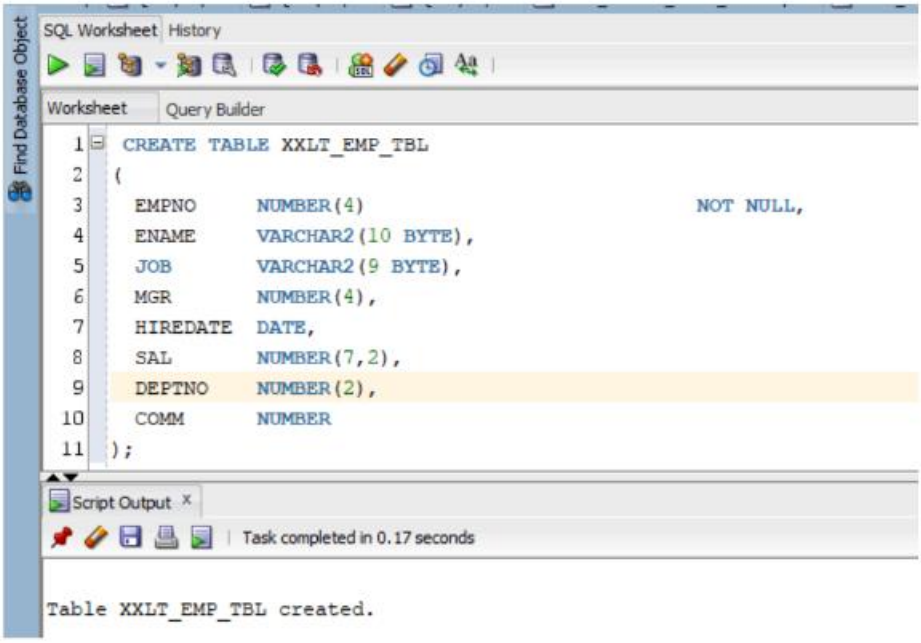
SQLLoader

Example where file name and path is sent as a parameter when registered as a concurrent program

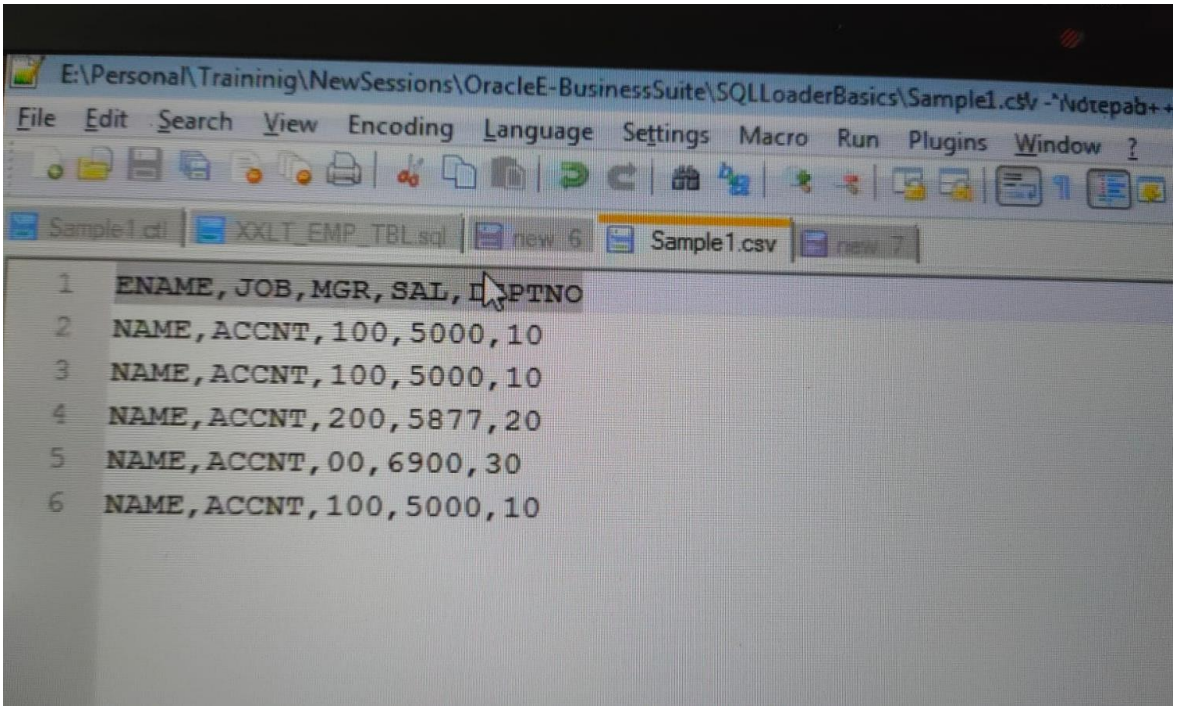
```
LOAD DATA

INFILE '$FILE'

INTO TABLE kap_emp

FIELDS TERMINATED BY ","

( emp_num, emp_name, department_num, department_name )
```

LOADING THE DATA USING CMD 'SQLLDR' EXAMPLE:

FIRST CREATE TABLE:



CREATE DATA FILE(SAMPLE1.CSV):



CREATE CONTROL FILE (SAMPLE1.CTL):

```
--/*****************************************************************************
-- *
-- *     Title                          : Sample1.ctl
-- *     Program type                   : CTL FILE
-- *     Description                    : Load data into XXLT_EMP_TBL
-- *                                      table from data file
-- *****************************************************************************
OPTIONS (SKIP = 1)
LOAD DATA
INFILE  Sample1.csv
INSERT
INTO TABLE XXLT_EMP_TBL
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
TRAILING NULLCOLS
(
  EMPNO    SEQUENCE(MAX,1) ,
  ENAME    "(:ENAME || :EMPNO)",
  JOB      ,
  MGR      ,
  HIREDATE SYSDATE,
  SAL      ,
  DEPTNO   ,
  COMM     CONSTANT 10
)
```

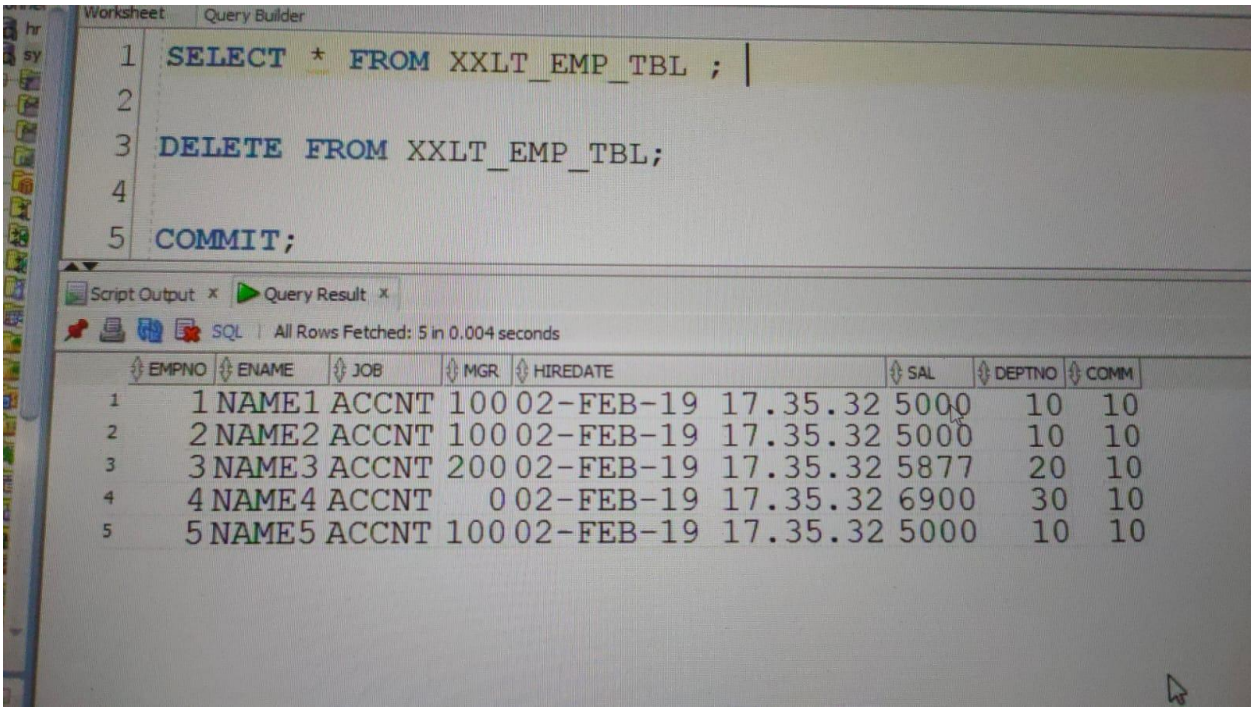
[OPTIONS (SKIP=1) TO SKIP THE FIRST ROW FROM .CSV FILE]

EMPNO SEQUENCE(MAX,1) ==> SEQUENCE IS FUNCTION IN SQLLOADER.

NOW USE THE CMD TO INSERT THE DATA IN THE TABLE:

```
C:\Windows\system32\cmd.exe
E:\Personal\Traininig\NewSessions\OracleE-BusinessSuite\SQLLoaderBasics>sqlldr control=Sample1.ctl userid=hr/hr
```

HR/HR ==> USERNAME/PASSWORD OF OUR SCHEMA



SQLLoader – Command Line Keywords

userid--Oracleusername/password

control--Controlfilename

log--Logfilename

bad--Badfilename

data--Datafilename

discard--Discardfilename

discardmax--Numberofdiscardstoallow (Defaultall)

skip--Numberoflogicalrecordstoskip (Default0)

load--Numberoflogicalrecordstoload (Defaultall)

errors--Numberoferrorstoallow (Default50)

PASSING DATA FILE NAME AT RUNTIME USING '$FILE' SO THAT FILE NAME WILL UPDATE BY ITSELF SO NO NEED TO HARDCODE FILENAME:

[INFILE '$FILE': You can use `INFILE '$FILE'` when the file path and name are **passed as parameters** to the control file, especially when running the SQL*Loader as a

concurrent program in Oracle EBS.The `$FILE` is typically a **substitution variable** that gets replaced with the actual file path at runtime.]

*C:\Users\onkar\Downloads\SQLLoaderBasics\Sample1.ctl - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

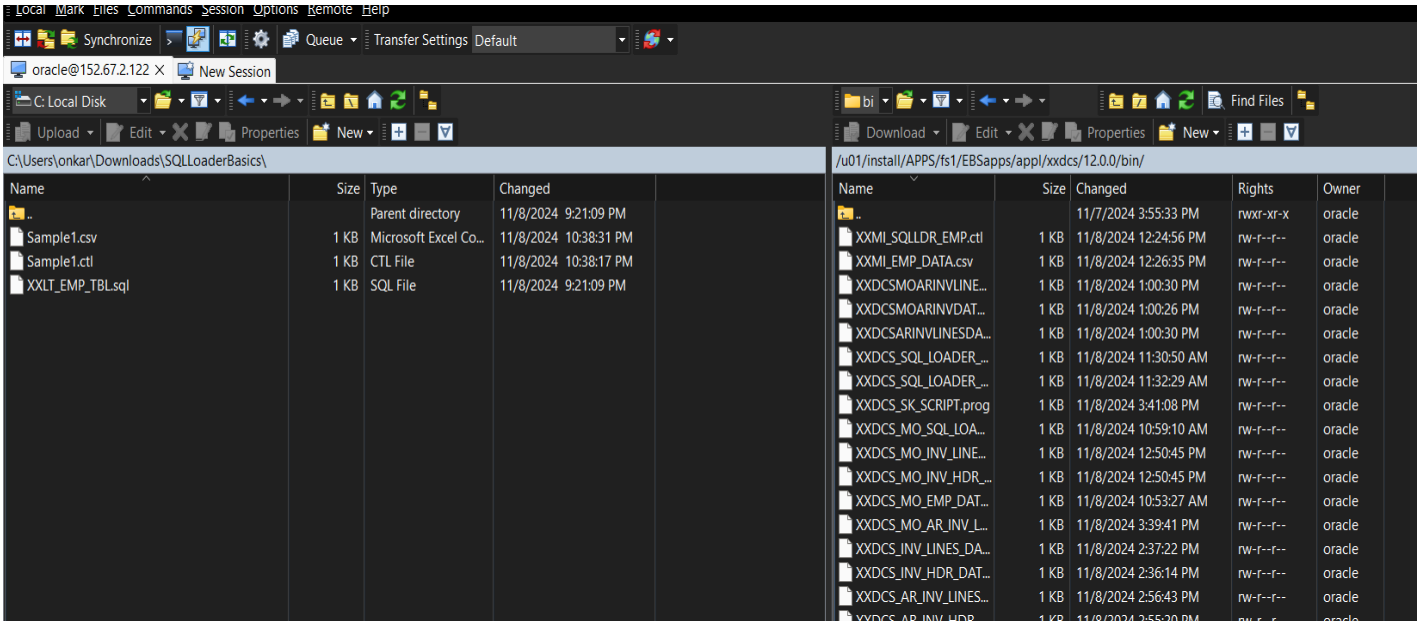
Sample1.ctl | Sample1.csv

```
 --/*************************************************************************
-- *
-- *    Title                          : Sample1.ctl
-- *    Program type                   : CTL FILE
-- *    Description                    : Load data into XXLT_EMP_TBL_ONKAR
-- *                                     table from data file
-- *************************************************************************
OPTIONS (SKIP = 1)
LOAD DATA
INFILE '$FILE'
APPEND
INTO TABLE XXLT_EMP_TBL_ONKAR
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
TRAILING NULLCOLS
(
  EMPNO    SEQUENCE(MAX,1) ,
  ENAME    "(:ENAME || :EMPNO)",
  JOB       ,
  MGR       ,
  HIREDATE SYSDATE,
  SAL       ,
  DEPTNO    ,
  COMM      CONSTANT 10
)
```

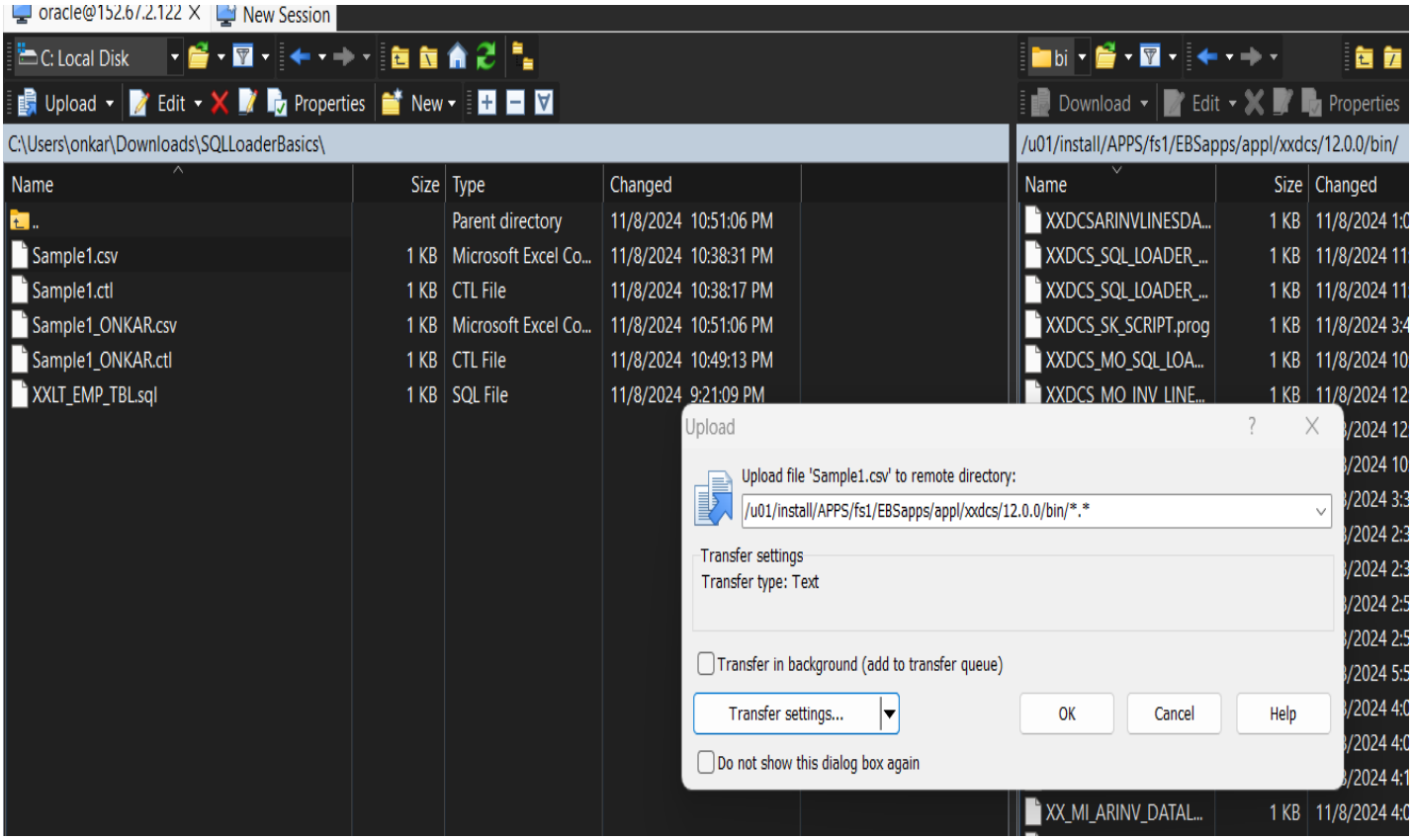
Sample1.ctl | Sample1.csv

```
ENAME,JOB,MGR,SAL,DEPTNO
NAME,ACCNT,100,5000,10
NAME,ACCNT,200,5877,20
```

MOVE THE DATA FILE(.CSV) IN TEXT FROMAT AND (.CTL) IN (.CTL) FORMAT FROM LOCAL TO SERVER:



TRANSFER SETTINGS ==> TEXT FORMAT



NOW CREATE EXECUTABLE,CONC PRO,REQ GRP AND RUN IT:

Concurrent Program Executable

Executable	ONKAR_SQLLOADER_CONC1
Short Name	ONKAR_SQLLOADER_CONC1
Application	xxdcs custom app
Description	
Execution Method	SQL*Loader
Execution File Name	Sample1_ONKAR
Subroutine Name	
Execution File Path	

Stage Function Parameters

Concurrent Programs

Program	ONKAR_SQLLOADER_CONC_PRO ☑ Enabled
Short Name	ONKAR_SQLLOADER_CONC_PRO
Application	xxdcs custom app
Description	

Executable

Name	ONKAR_SQLLOADER_CONC1	Options	
Method	SQL*Loader	Priority	

Request

Type			
Incrementor			
MLS Function			
Operating Unit Mode		Restart Mode	Restartable

☐ Allow Disabled Values ☑ Use in SRS
☑ NLS Compliant ☐ Enable Trace
☐ Recalculate Default Parameters ☐ Run Alone

Output

Format	Text
	☑ Save (C) ☑ Print
Columns	
Rows	
Style	
	☐ Style Required
Printer	

Propagation Of Options to Children

☐ Delivery Options ☐ Notifications

Security

☐ Disable URL Security
Maximum Output URL Views

Business Events

☐ Request Submitted (Y) ☐ Program Completed ☐ Request On Hold ☐ Post Processing Started
☐ Request Running ☐ Request Completed (Z) ☐ Request Resumed ☐ Post Processing Ended

Copy to... | Session Control | Incompatibilities | Parameters

Concurrent Program Parameters

Program: ONKAR_SQLLOADER_CONC_PRO
Application: xxdcs custom app

Conflicts Domain:
Security Group:

Seq	Parameter	Description	Enabled
1	ENTER FILE NAME		☑

Validation

Value Set: 100 Characters
Default Type:
☑ Allow Update
☐ Required
☐ Enable Security
Description: 100 Characters
Default Value:
Range:

☑ Display

Display Size: 50
Description Size: 50
Prompt: ENTER FILE NAME
Concatenated Description Size: 25
LOV Prompt: ENTER FILE NAME

Token:

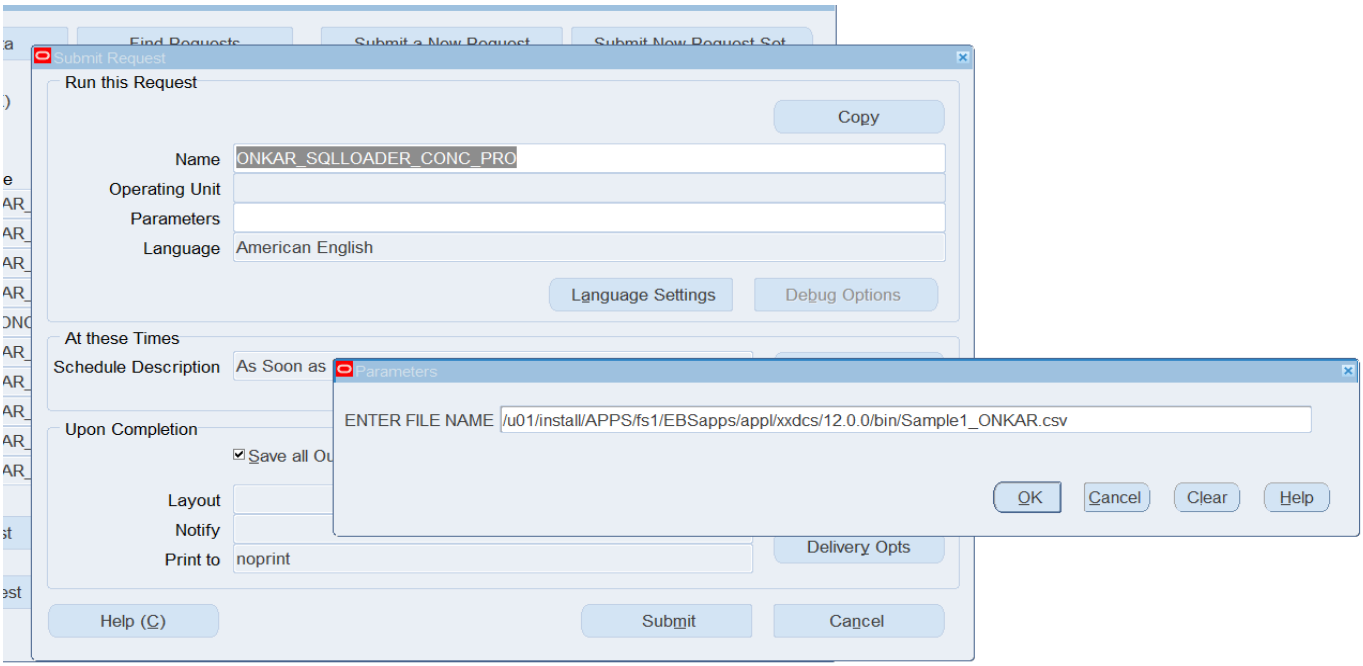
Request Groups

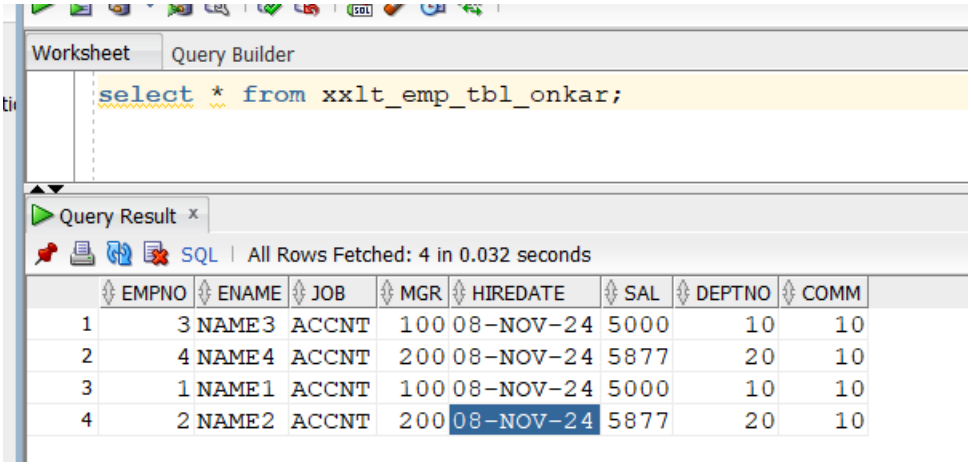
Group: XXDCS
Application: xxdcs custom app
Code: XXDCS
Description:

Requests

Type	Name	Application
Program	XXDCS_PO_HEADER_DETL	xxdcs custom app
Program	XXDCS_NEW_TEST	xxdcs custom app
Program	ONKAR_SQLLOADER_CONC_PRO	xxdcs custom app

Description:





	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	DEPTNO	COMM
1	3	NAME3	ACCNT	100	08-NOV-24	5000	10	10
2	4	NAME4	ACCNT	200	08-NOV-24	5877	20	10
3	1	NAME1	ACCNT	100	08-NOV-24	5000	10	10
4	2	NAME2	ACCNT	200	08-NOV-24	5877	20	10

Successfully inserted 2 rows.

(Request runs 2 times thus 4 rows we got)