# Process Overview

A process is required to import data in CSV files into the database.

1. The files are named with the extension ‘csv’.
2. Each file must have a unique name.
3. The process must be automated to run every hour, and import all new CSV files received.
4. The data must be validated, and if there are any errors, the entire CSV file must be rejected, moved to an ‘Error’ directory, and all errors reported.
5. A manual procedure is required to fix rejected files.
6. If the data passes validation, load the data into the associated database tables.
7. Move successfully imported files to a ‘Processed’ archive directory.
8. Delete old error messages relating to the orders that have been successfully imported.

## Fixing rejected files

If a single error is found when validating the data, the entire CSV file is rejected, and no data is loaded into the database. The errors in the CSV file must be fixed, and the file must be re-processed. It is much easier to fix all the errors in the file and re-import the whole thing, than to try and pick out the records that need fixing from a large file.

1. Manual intervention is required.
2. An import error report will show the following information for each rejected file:
   1. Filename.
   2. A key value that uniquely identifies the data which has errors.
   3. CSV data record (all fields).
   4. An error message identifying the invalid data.
   5. Date and time error reported.
3. Locate the rejected CSV file in the Error directory, via the filename on the error report.
4. Edit the CSV file, and manually correct each error reported.
5. Move the corrected file to the ‘Received’ directory to be re-processed.

# Database Tables

Determine which tables you will load the CSV data into. The DEMO table will be used as an example.

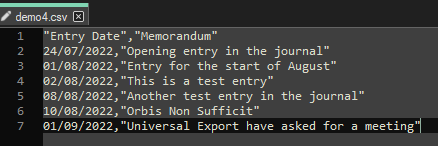
|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Data Type** | **Size** | **Comments** |
| ENTRY\_DATE | DATE |  |  |
| MEMORANDUM | VARCHAR2 | 4000 |  |

# CSV file format

Determine the format of the CSV file containing the data to be imported.

|  |  |
| --- | --- |
| **Record Type** | **Description** |
| Header | Header record with field names in double quotes |
| Body | CSV data records, fields separated by commas. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field No.** | **Name** | **Data Type** | **Size** | **Description** |
| 1 | ENTRY\_DATE | Date | 10 | Memo entry date, format DD/MM/YYYY |
| 2 | MEMORANDUM | Char | 4000 | Memo text |



# Directories

|  |  |
| --- | --- |
| **Directory Name** | **Description** |
| DATA\_IN\received | CSV files received here |
| DATA\_IN | Move files here for processing |
| DATA\_IN\_PROCESSED | Successfully imported files moved here |
| DATA\_IN\_ERROR | If errors found during validation move CSV file here |

# Modules

You will need to create the following program modules, named appropriately for the data being imported.

|  |  |
| --- | --- |
| **Program Name** | **Description** |
| IMPORT\_<name>.BAT | DOS Batch script. Finds CSV files. Moves file to DATA\_IN. Runs SQL\*Plus script IMPORT\_<name>.SQL to process data. |
| IMPORT\_<name>.SQL | SQL script calls a package function to validate the CSV data |
| IMPORT.<name>\_IMP | PL/SQL package function. Validates the CSV data, reports errors in IMPORTERROR table. If no errors, loads order into database, moves CSV file to DATA\_IN\_PROCESSED. |

## IMPORT\_<name>.BAT

DOS Batch script.

Search the received directory for CSV files containing data to import.

For each CSV file found with the name ‘\*.csv’:

Copy the CSV file to the DATA\_IN import directory.

Run PL/SQL: execute script IMPORT\_<name>.SQL passing filename.

Delete the CSV file from the received directory.

## IMPORT\_<name>.SQL

Call PL/SQL package function **IMPORT.<name>\_IMP** passing filename.

## IMPORT.<name>\_IMP

1. Call the package function UTIL\_FILE.LOAD\_CSV to load data from a CSV file into the IMPORTCSV staging table.
   1. The load\_csv function returns an integer FILEID, which identifies the group of records loaded from the CSV file into the staging table.
2. If the file was not found, report error and stop processing.
3. Validate the data in IMPORTCSV matching FILEID.
   1. Set field KEY\_VALUE in IMPORTCSV to a unique value, that identifies each record, or group of records. In this case it will be the first field in the CSV file, ENTRY\_DATE.
   2. Record all validation errors found in the IMPORTERROR table, including the KEY\_VALUE field.
4. If data fails validation:
   1. Delete the data from the IMPORTCSV staging table.
   2. Move the CSV file to the error directory.
   3. Stop processing, exit with an error status.
5. If data passes validation:
   1. Insert data into the database tables.
   2. Delete old error messages from the IMPORTERROR table for the data successfully imported, using the KEY\_VALUE column of IMPORTCSV.
   3. Delete the data from the IMPORTCSV staging table.
   4. Move the CSV file to the processed directory.
   5. Exit with a success status.

## PL/SQL Packages

### Template Code

The following demo code can be used as a template for creating each CSV import process.

|  |  |
| --- | --- |
| **Name** | **Description** |
| IMPORT.DEMO\_VALID | Validate the data in the CSV file, report errors |
| IMPORT.DEMO\_IMP | Import the demo data. Load CSV into staging table, validate, and populate database tables. |

### Generic Procedures and Functions

Note that the following functions and procedures are generic, and can be re-used when creating each additional data import process.

|  |  |
| --- | --- |
| **Name** | **Description** |
| IMPORT.DELETE\_ERROR | Delete old error messages for previously failed imports. |
| IMPORT.IMPORT\_ERROR | Record validation error message on IMPORTERROR table |
| UTIL\_FILE.LOAD\_CSV | Load CSV file data into the IMPORTCSV table |
| UTIL\_FILE.DELETE\_CSV | Delete rows from IMPORTCSV for each CSV file that has been processed |
| UTIL\_FILE.RENAME\_FILE | Rename the CSV file by moving it to directory DATA\_IN\_PROCESSED if order imported, or DATA\_IN\_ERROR if it failed validation |
| UTIL\_STRING.GET\_FIELD | Extract Nth field from a delimited string (the CSV record) |
| UTIL\_ADMIN.LOG\_MESSAGE | Record errors in the application log table APPLOG |

### Validation

A function is required to validate the CSV data in the staging table column IMPORTCSV.CSV\_REC. Record all validation errors in the IMPORTERROR table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Number** | **Description** | **Data Type** | **Validation Rule** |
| 1 | Entry Date | Date | Valid date format DD/MM/YYYY |
| 2 | Memorandum | String | Length must not exceed 4000 characters |

### Tables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table Name** | **Select** | **Insert** | **Update** | **Delete** |
| APPLOG |  | X |  |  |
| IMPORTCSV | X | X | X | X |
| IMPORTERROR |  | X |  | X |
| DEMO | X | X | X |  |

#### IMPORTCSV

Description: Staging table, data from each CSV file is loaded into this table to be validated

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Data Type** | **Size** | **Comments** |
| RECID | NUMBER | 28 | Unique sequential primary index |
| FILEID | NUMBER | 28 | Unique ID for each CSV file |
| FILENAME | VARCHAR2 | 255 | CSV filename |
| CSV\_REC | VARCHAR2 | 4000 | CSV data record |
| KEY\_VALUE | VARCHAR2 | 30 | Field of CSV\_REC that identifies data being imported. Update this column during validation. |

#### IMPORTERROR

Description: Report all import validation errors in this table, which will be used to generate an error report. A row will be inserted on this table for each field that is invalid.

|  |  |  |  |
| --- | --- | --- | --- |
| **Column** | **Data Type** | **Size** | **Comments** |
| RECID | NUMBER | 28 | Unique sequential primary index |
| FILENAME | VARCHAR2 | 255 | CSV filename |
| ERROR\_DATA | VARCHAR2 | 4000 | CSV record that has failed validation |
| ERROR\_MESSAGE | VARCHAR2 | 1000 | Error message identifying invalid field |
| ERROR\_TIME | TIMESTAMP | 6 | Date and Time error logged |
| USER\_NAME | VARCHAR2 | 128 | Database user who ran import process |
| KEY\_VALUE | VARCHAR2 | 30 | Field of CSV\_REC that identifies data being imported. |
| IMPORT\_SQLERRM | VARCHAR2 | 1000 | SQLERRM error message |

#### DEMO

The data from the CSV file will be loaded into the demo database table.

|  |  |  |
| --- | --- | --- |
| **Column** | **Maps to CSV field** | **Notes** |
| ENTRY\_DATE | Field 1 |  |
| MEMORANDUM | Field 2 |  |