# Import Order CSV Files

# Requirements

A process is required to import sales order data into the database.

1. The sales order data is received in CSV format text files.
2. The files are named with the prefix ‘order’ and extension ‘csv’.
3. Each file must have a unique name.
4. The process must be automated to run every hour, and import all new order CSV files received.
5. 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.
6. A manual procedure is required to fix rejected files.
7. If the data passes validation, load the data into the order tables.
8. Move successfully imported files to a ‘Processed’ archive directory.
9. Delete old error messages relating to the orders that have been successfully imported.

## Fixing rejected files

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 order: Order Reference.
	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.
	1. Order Reference may be altered, but you must check that there are no existing orders with the same reference.
	2. Dates must be in the format DD/MM/YYYY.
	3. Ship Date must be on or later than the Order Date, in the format DD/MM/YYYY.
	4. Customer ID invalid: contact the sales department to either obtain the correct code, or have a new account created.
	5. Product ID invalid: contact the sales department for the correct code.
5. Move the corrected file to the ‘Received’ directory to be re-processed.

# Technical Design

## CSV File format

|  |  |
| --- | --- |
| **Record Type** | **Description** |
| Header  | Header record with field names in double quotes |
| Body | Order data records. One record per line of the order.  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field No.** | **Name** | **Data Type** | **Size** | **Description** |
| 1 | Ord Ref | Char | 10 | Order Reference. Maximum length 10 characters. Must be unique per order. |
| 2 | Order Date | Date | 10 | Date format DD/MM/YYYY |
| 3 | Commission | Char | 1 | A to Z, may be null |
| 4 | Customer ID | Number | 6 | Must exist on the Customer table |
| 5 | Ship Date | Date | 10 | Date format DD/MM/YYYY. Must be on or later than the Order Date |
| 6 | Product ID | Number | 6 | Must exist on the Product table |
| 7 | Qty | Number | 8 | Quantity of product ordered. Must be a valid whole number > 0, and <= 99999999. |

### Sample CSV Data



## 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

|  |  |
| --- | --- |
| **Program Name** | **Description** |
| IMPORT\_ORDER.BAT | DOS Batch script. Finds order CSV files. Moves file to DATA\_IN. Runs SQL\*Plus script IMPORT\_ORDER.SQL to process data. |
| IMPORT\_ORDER.SQL | SQL script calls a package function to validate the CSV data |
| IMPORT.ORD\_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\_ORDER.BAT

DOS Batch script.

Search the received directory for CSV files containing order data.

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

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

 Run PL/SQL: execute script IMPORT\_ORDER.SQL passing filename.

 Delete the CSV file from the received directory.

## IMPORT\_ORDER.SQL

Call PL/SQL package function **IMPORT.ORD\_IMP** passing filename.

## IMPORT.ORD\_IMP

1. Call the package function UTIL\_FILE.LOAD\_CSV to load order 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 column KEY\_VALUE in table IMPORTCSV to a unique value, that identifies each order, in this case it will be the first field in the CSV file, ORDREF.
	2. Record all validation errors found in the IMPORTERROR table, including the KEY\_VALUE column.
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 ORD and ITEM tables.
	2. Delete old error messages from the IMPORTERROR table for the orders 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 Functions and Procedures

The package function ORD\_VALID will need to be created. The other functions/procedures already exist, for use as described.

|  |  |  |
| --- | --- | --- |
| **Name** |  | **Description** |
| IMPORT.ORD\_VALID |  | Validate the order data in the CSV file, report errors |
| IMPORT.DELETE\_ERROR | \* | Delete old error messages for orders that have been successfully imported |
| 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 |
| ORDERRP.CURRENTPRICE | \* | Finds the current price for the specified product |

Note that the above functions and procedures marked with \* are generic, and can be re-used when creating additional data import processes.

### 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 | Ord Ref (Key Value) | Char | Length <= 10 characters. Must not already exist on Ord (duplicate ORDREF) |
| 2 | Order Date | Date | Valid date format DD/MM/YYYY |
| 3 | Comm Plan | Char | Must be a single character |
| 4 | Customer ID | Number | Must find row on Customer with matching CUSTID |
| 5 | Ship Date | Date | Valid date format DD/MM/YYYY. Must be on, or a later date than Order Date |
| 6 | Product ID | Number | Must find row on Product with matching PRODID |
| 7 | Quantity | Number | 0 to 99999999 |

### Tables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table Name** | **Select** | **Insert** | **Update** | **Delete** |
| APPLOG |  | X |  |  |
| IMPORTCSV | X | X | X | X |
| IMPORTERROR |  | X |  | X |
| ORD | X | X | X |  |
| ITEM |  | X |  |  |
| PRODUCT | X |  |  |  |
| PRICE | X |  |  |  |
| CUSTOMER | 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 | ORDREF, first field of CSV\_REC identifies each order to be 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 | ORDREF, first field of CSV\_REC identifies each order to be imported |
| IMPORT\_SQLERRM | VARCHAR2 | 1000 | SQLERRM error message |

#### ORD

Insert at change of Ord Ref (Key Value)

|  |  |  |
| --- | --- | --- |
| **Column** | **Maps to CSV field** | **Notes** |
| ORDID | N/A | Generate next value from ORDID\_SEQ |
| ORDREF | Field 1 |  |
| ORDERDATE | Field 2 |  |
| COMMPLAN | Field 3 |  |
| CUSTID | Field 4 |  |
| SHIPDATE | Field 5 |  |
| TOTAL |  | Calculated as total of ITEM.ITEMTOT for each order |

#### ITEM

Insert for each row on IMPORTCSV

|  |  |  |
| --- | --- | --- |
| **Column** | **Maps to CSV field** | **Notes** |
| ORDID | N/A | Generate next value from ORDID\_SEQ |
| ITEMID | N/A | Reset to 1 at change of ORDREF, increment by 1 for each ITEM row inserted |
| PRODID | Field 6 |  |
| ACTUALPRICE |  | Find on PRICE. PRODID = Field 6, STARTDATE on or before current date, ENDDATE on or after current date |
| QTY | Field 7 |  |
| ITEMTOT |  | = QTY \* ACTUALPRICE |