Importing Data into Your App from Excel, XML or CSV files

Data may be gathered for importion into your App from any outside source. Automating the importation process so users do not have to labouriously enter data can save valuable resources.

Due to the complexity of Orixa data the importation process often requires careful work and thought to ensure that the relational connections built into the database are preserved during the importation process.

For a one-off importation process the Developer can craft a single SQL Statement using the IMPORT TABLE SQL Keywords. For more frequent data-importation processes the Developer may wish to write a SQL Procedure which other users can run.

Before you begin

1. Find the data you wish to import and if the process is to be run repeatedly ensure you have a set of steps that will always produce data with exactly the same format. A data importation procedure will fail if the source CSV File is not correctly formatted to its expectations.
2. Review how you will generate the Orixa IDs or Codes in the CSV file that allow you to link data together in your App. It may be that users creating the CSV File must be given a reference file, containing IDs for them to use, or some other link-field must be used.

Importing using the IMPORT TABLE SQL Statement

In simple and one-off cases the Developer can write SQL to import data. This actually works in a similar way to the CSV Importation Utility, but as the Developer can also create temporary tables and run multiple SQL Statements allows more flexibility.

```
IMPORT TABLE [Table or View Name]
FROM [FileName]
IN STORE [StoreName]
[([ColumnName][,[ColumnName]])]
[FORMAT DELIMITED|XML]
[ENCODING AUTO|ANSI|UNICODE]
[DELIMITER CHAR [DelimiterChar]]
[QUOTE CHAR [QuoteChar]]
[DATE FORMAT [DateFormat]]
[TIME FORMAT [TimeFormat] [AM LITERAL [AMLiteral] PM LITERAL [PMLiteral]]]
[DECIMAL CHAR [DecimalChar]]
[BOOLEAN TRUE LITERAL [TrueLiteral] FALSE LITERAL [FalseLiteral]]
[USE HEADERS]
[MAX ROWS [MaxRowCount]]
```

Example of the IMPORT TABLE Statement

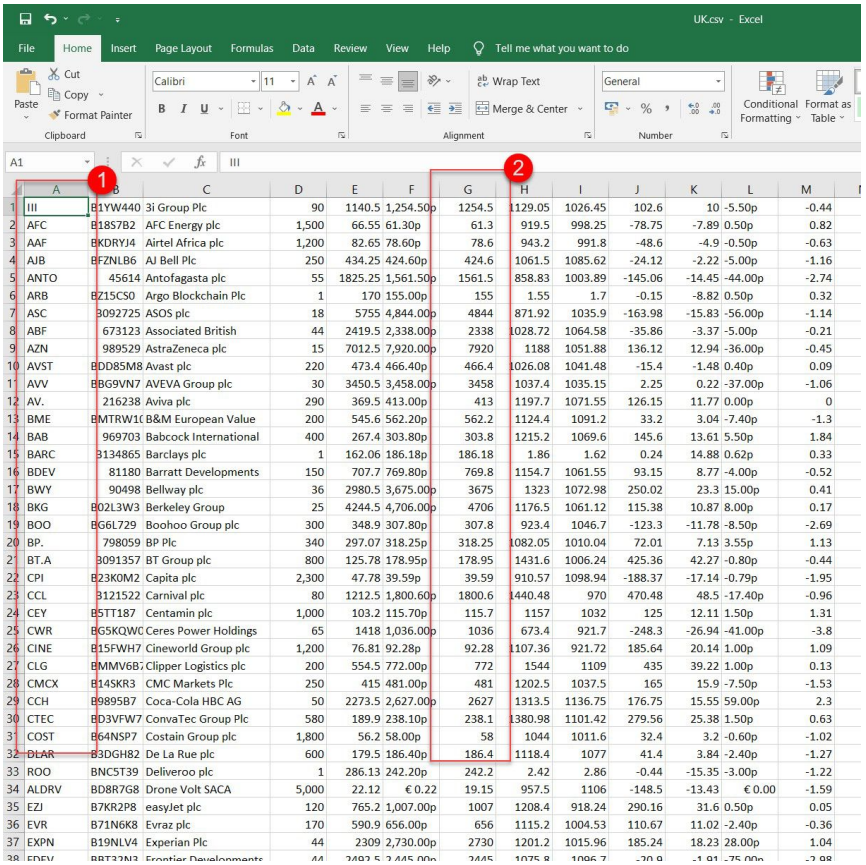
```
IMPORT TABLE "Valuations"
FROM "ImportsJune2021.csv"
IN STORE "Imports"
```

Note from the above that it is rarely necessary to use the additional keyword of the IMPORT TABLE SQL Statement, if the CSV file is well formed.

Worked Example: Importing Stock Valuations from an Online source into an Orixa App

Orixa has been used to make a Pension Fund Management App. The Pension Fund managers buy stocks and shares for their fund. The Fund managers want to keep records of the variation of stock-values over time so they can assess which stocks to buy.

The following example shows how they have done this. Your Developer can replicate this process within your Orixa App with other data sources.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	III	B1YW440	3i Group Plc	90	1140.5	1,254.50p	1254.5	1129.05	1026.45	102.6	10	-5.50p	-0.44
2	AFC	B18S7B2	AFC Energy plc	1,500	66.55	61.30p	61.3	919.5	998.25	-78.75	-7.89	0.50p	0.82
3	AAF	BKDRYJ4	Airtel Africa plc	1,200	82.65	78.60p	78.6	943.2	991.8	-48.6	-4.9	-0.50p	-0.63
4	AJB	BFZNLB6	AJ Bell Plc	250	434.25	424.60p	424.6	1061.5	1085.62	-24.12	-2.22	-5.00p	-1.16
5	ANTO	45614	Antofagasta plc	55	1825.25	1,561.50p	1561.5	858.83	1003.89	-145.06	-14.45	-44.00p	-2.74
6	ARB	BZ15CS0	Argo Blockchain Plc	1	170	155.00p	155	1.55	1.7	-0.15	-8.82	0.50p	0.32
7	ASC	3092725	ASOS plc	18	5755	4,844.00p	4844	871.92	1035.9	-163.98	-15.83	-56.00p	-1.14
8	ABF	673123	Associated British	44	2419.5	2,338.00p	2338	1028.72	1064.58	-35.86	-3.37	-5.00p	-0.21
9	AZN	989529	AstraZeneca plc	15	7012.5	7,920.00p	7920	1188	1051.88	136.12	12.94	-36.00p	-0.45
10	AVST	BDD85M8	Avast plc	220	473.4	466.40p	466.4	1026.08	1041.48	-15.4	-1.48	0.40p	0.09
11	AVV	BBG9VN7	AVEVA Group plc	30	3450.5	3,458.00p	3458	1037.4	1035.15	2.25	0.22	-37.00p	-1.06
12	AV.	216238	Aviva plc	290	369.5	413.00p	413	1197.7	1071.55	126.15	11.77	0.00p	0
13	BME	BMTRW1C	B&M European Value	200	545.6	562.20p	562.2	1124.4	1091.2	33.2	3.04	-7.40p	-1.3
14	BAB	969703	Babcock International	400	267.4	303.80p	303.8	1215.2	1069.6	145.6	13.61	5.50p	1.84
15	BARC	3134865	Barclays plc	1	162.06	186.18p	186.18	1.86	1.62	0.24	14.88	0.62p	0.33
16	BDEV	81180	Barratt Developments	150	707.7	769.80p	769.8	1154.7	1061.55	93.15	8.77	-4.00p	-0.52
17	BWY	90498	Bellway plc	36	2980.5	3,675.00p	3675	1323	1072.98	250.02	23.3	15.00p	0.41
18	BKG	B02L3W3	Berkeley Group	25	4244.5	4,706.00p	4706	1176.5	1061.12	115.38	10.87	8.00p	0.17
19	BOO	BG6L729	Boohoo Group plc	300	348.9	307.80p	307.8	923.4	1046.7	-123.3	-11.78	-8.50p	-2.69
20	BP.	798059	BP Plc	340	297.07	318.25p	318.25	1082.05	1010.04	72.01	7.13	3.55p	1.13
21	BT.A	3091357	BT Group plc	800	125.78	178.95p	178.95	1431.6	1006.24	425.36	42.27	-0.80p	-0.44
22	CPI	B23K0M2	Capita plc	2,300	47.78	39.59p	39.59	910.57	1098.94	-188.37	-17.14	-0.79p	-1.95
23	CCL	3121522	Carnival plc	80	1212.5	1,800.60p	1800.6	1440.48	970	470.48	48.5	-17.40p	-0.96
24	CEY	B5TT187	Centamin plc	1,000	103.2	115.70p	115.7	1157	1032	125	12.11	1.50p	1.31
25	CWR	BG5KQWC	Ceres Power Holdings	65	1418	1,036.00p	1036	673.4	921.7	-248.3	-26.94	-41.00p	-3.8
26	CINE	B1FWH7	Cineworld Group plc	1,200	76.81	92.28p	92.28	1107.36	921.72	185.64	20.14	1.00p	1.09
27	CLG	BMMV6B7	Clipper Logistics plc	200	554.5	772.00p	772	1544	1109	435	39.22	1.00p	0.13
28	CMCX	B14SKR3	CMC Markets Plc	250	415	481.00p	481	1202.5	1037.5	165	15.9	-7.50p	-1.53
29	CCH	B9895B7	Coca-Cola HBC AG	50	2273.5	2,627.00p	2627	1313.5	1136.75	176.75	15.55	59.00p	2.3
30	CTEC	BD3VFW7	ConvaTec Group Plc	580	189.9	238.10p	238.1	1380.98	1101.42	279.56	25.38	1.50p	0.63
31	COST	B64NSP7	Costain Group plc	1,800	56.2	58.00p	58	1044	1011.6	32.4	3.2	-0.60p	-1.02
32	DLAR	B3DGH82	De La Rue plc	600	179.5	186.40p	186.4	1118.4	1077	41.4	3.84	-2.40p	-1.27
33	ROO	BNC5T39	Deliveroo plc	1	286.13	242.20p	242.2	2.42	2.86	-0.44	-15.35	-3.00p	-1.22
34	ALDRV	BD8R7G8	Drone Volt SACA	5,000	22.12	€ 0.22	19.15	957.5	1106	-148.5	-13.43	€ 0.00	-1.59
35	EZJ	B7KR2P8	easyJet plc	120	765.2	1,007.00p	1007	1208.4	918.24	290.16	31.6	0.50p	0.05
36	EVR	B71N6K8	Evraz plc	170	590.9	656.00p	656	1115.2	1004.53	110.67	11.02	-2.40p	-0.36
37	EXPN	B19NLV4	Experian Plc	44	2309	2,730.00p	2730	1201.2	1015.96	185.24	18.23	28.00p	1.04

CSV File with Stock Values

They are sent a daily CSV file containing hundreds of stock values. This file comes in a standard, simple format they know will always be the same, as shown in the image above.

> 1. The first column of the CSV File contains the Stock "Symbol", a unique code that identifies the stock.
> 2. Once of the other columns ("G" in this case) contains "today's price" for this stock.

Their Orixa App contains a "Valuations" data-table, with the following structure:

```
CREATE TABLE "Valuations"
( "ID" INTEGER DEFAULT UID() NOT NULL,
"StocksID" INTEGER DESCRIPTION 'DefaultRI',
"DateDone" DATE DEFAULT CURRENT_DATE,
"Price" DECIMAL(19,4),
"DateCreated" TIMESTAMP DEFAULT Current_Timestamp,
"Complete" BOOLEAN DEFAULT false,
CONSTRAINT "PrimaryKey" PRIMARY KEY ("ID"),
CONSTRAINT "StocksID" FOREIGN KEY ("StocksID") REFERENCES "Stocks" ("ID")
DESCRIPTION 'Parent' )
```

Note that this data-table contains a "StocksID" and a price. To import the data all that is needed is to copy in the Price data, together with a date, and the StocksID. However, the StocksID is **not** the same as the Stock "Symbol"

Their Orixa App contains a "Stocks" data-table, with the following structure:

```
CREATE TABLE "Stocks"
( "ID" INTEGER DEFAULT UID() NOT NULL,
"Symbol" VARCHAR(10) COLLATE "ANSI",
"Name" VARCHAR(60) COLLATE "ANSI" NOT NULL,
"Description" CLOB COLLATE "ANSI",
[-table definition continues-]
```

To make the importation work write a procedure tα

> 1. Import all the data from the CSV file into a **temporary** data-table. Note that the procedure must first test whether the temporary table already exists, and call DROP to remove it if it does.
> 2. Run an INSERT statement which coalesces the data from the temporary data-table with data from the

> 2. "Stocks" data-table, to match the Symbol and link it to the StocksID.

Full listing of the "ImportStockValuations" procedure

```
PROCEDURE "ImportStockValuations" (IN "aCSVFileName" VARCHAR(120) COLLATE "ANSI", IN "aDateDone"
DATE)
BEGIN
  DECLARE Crsr CURSOR WITH RETURN FOR Stmt;
  DECLARE Names CLOB;
 DECLARE Symbol VARCHAR;
  DECLARE Stock VARCHAR;
--check existence of temp table and drop if present
PREPARE Stmt FROM
' SELECT * FROM Information.TemporaryTables
  WHERE Name = ''ImportValuations'' ';
OPEN Crsr;
IF ROWCOUNT(Crsr) > 0 THEN
  EXECUTE IMMEDIATE
  ' DROP TABLE "ImportValuations"';
  END IF;
--create temp-table
EXECUTE IMMEDIATE
' CREATE TEMPORARY TABLE "ImportValuations"
("Code" VARCHAR(10) COLLATE "ANSI",
"Sedol" VARCHAR(50) COLLATE "ANSI",
"Stock" VARCHAR(100) COLLATE "ANSI",
"Units" VARCHAR(50),
"UnitCost" VARCHAR(50),
"Price" VARCHAR(50),
"PriceP" VARCHAR(50),
"ValueP" VARCHAR(50),
"CostP" VARCHAR(50),
"GL" VARCHAR(50),
"GLPercent" VARCHAR(50),
"DayChange" VARCHAR(50),
"DayChangePercent" VARCHAR(50)) ';

EXECUTE IMMEDIATE
' SET FILES STORE TO "Imports" ';

--find the proposed import file, to check the user has the name right
PREPARE Stmt FROM
' SELECT LIST(UPPER(Name)) as Names FROM Configuration.Files
  WHERE UPPER(Name) LIKE ''%.CSV'' ';
OPEN Crsr;
FETCH FIRST FROM Crsr('Names') INTO Names;
IF POSITION(UPPER(aCSVFilename), ' ' + Names) > 0 THEN
  EXECUTE IMMEDIATE
  ' IMPORT TABLE "ImportValuations"
    FROM "' + aCSVFileName + '"
    IN STORE "Imports"
    FORMAT DELIMITED ';
  EXECUTE IMMEDIATE
  ' RENAME FILE "' + aCSVFileName + '"
    IN STORE "Imports" TO "' + REPLACE('csv' WITH 'OLD' IN aCSVFileName) + '"';
  PREPARE Stmt FROM
    ' SELECT
        Code,
        REPLACE('''''''' WITH '''"''' IN Stock) AS "Stock"
      FROM ImportValuations
      WHERE NOT Code IN ( SELECT Symbol FROM Stocks ) ';
  OPEN Crsr;
  FETCH FIRST FROM Crsr('Code', 'Stock') INTO Symbol, Stock;
```

```
  WHILE NOT EOF(Crsr) DO
    IF ("Symbol" IS NOT NULL) AND ("Symbol" <> '') THEN
      EXECUTE IMMEDIATE
      ' INSERT INTO Stocks
        (Symbol, Name, WatchListID)
        VALUES
        (''' + Symbol + ''', CAST(''' + Stock + ''' AS VARCHAR(60)), 22168 ) ';
      FETCH NEXT FROM Crsr('Code', 'Stock') INTO Symbol, Stock;
      END IF;
    END WHILE;
  EXECUTE IMMEDIATE
  ' INSERT INTO Valuations
    (StocksID, DateDone, Price)
    SELECT
    S.ID,
    DATE ''' + CAST(aDateDone AS VARCHAR) + ''',
    CAST(I.PriceP as FLOAT) / 100
    FROM ImportValuations I
    LEFT JOIN Stocks S ON S.Symbol = I.Code
    WHERE I.Code IN
    (SELECT Symbol FROM Stocks) ';
  END IF;
END
```

What the different parts of the procedure do

`PREPARE Stmt FROM`
`' SELECT * FROM Information.TemporaryTables`
`WHERE Name = ''ImportValuations'' ';`
`OPEN Crsr;`
`IF ROWCOUNT(Crsr) > 0 THEN`
`  EXECUTE IMMEDIATE`
`  ' DROP TABLE "ImportValuations"';`
`  END IF;`
	Open a statement which tests whether the "ImportValuations" temporary table already exists. If it does DROP it.

`EXECUTE IMMEDIATE`
`' CREATE TEMPORARY TABLE "ImportValuations"`
`("Code" VARCHAR(10) COLLATE "ANSI",`
`"Sedol" VARCHAR(50) COLLATE "ANSI",`
`"Stock" VARCHAR(100) COLLATE "ANSI",`
`"Units" VARCHAR(50),`
`"UnitCost" VARCHAR(50),`
`"Price" VARCHAR(50),`
`"PriceP" VARCHAR(50),`
`"ValueP" VARCHAR(50),`
`"CostP" VARCHAR(50),`
`"GL" VARCHAR(50),`
`"GLPercent" VARCHAR(50),`
`"DayChange" VARCHAR(50),`
`"DayChangePercent" VARCHAR(50)) ';`
	Run a statement to CREATE the "ImportValuations" temporary table.
	
	Important: Note that the structure of this temporary table exactly matches the number of columns used in the CSV file.
	
	Also: Note that data-types such as "FLOAT" and "DECIMAL" are NOT used. All data in the temporary table is imported as "VARCHAR" to avoid possible issues with NULL values.

`EXECUTE IMMEDIATE`
`' IMPORT TABLE "ImportValuations"`
`  FROM "' + aCSVFileName + '"`
`  IN STORE "Imports"`
`  FORMAT DELIMITED ';`
`EXECUTE IMMEDIATE`
`  ' RENAME FILE "' + aCSVFileName + '"`
`IN STORE "Imports" TO "' + REPLACE('csv' WITH`
	Actually call the "IMPORT TABLE" SQL statement

```
'OLD' IN aCSVFileName) + '"';
```

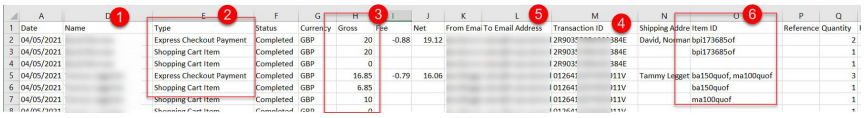
```
EXECUTE IMMEDIATE
  ' INSERT INTO Valuations
     (StocksID, DateDone, Price)
  SELECT
     S.ID,
     DATE ''' + CAST(aDateDone AS VARCHAR) +
''',
     CAST(I.PriceP as FLOAT) / 100
     FROM ImportValuations I
     LEFT JOIN Stocks S ON S.Symbol = I.Code
     WHERE I.Code IN
     (SELECT Symbol FROM Stocks) ';
```

This section of the procedure actually INSERTs data.

Note how the ImportValuations Temporary table is JOINed to the permanent "Stocks" data-table to ensure that the Valuations data is imported **with the correct StocksID.**

Worked Example: Importing Paypal Online Sales Transaction Data into an Orixa App

In this example a business sells their products online using Paypal as the transaction management system to handle payments from customers. A procedure has been written to import data from Paypal directly into the Orixa App to avoid re-keying sales information.



	A	B	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Date	Name	Type	Status	Currency	Gross	Fee	Net	From Emai	To Email Address	Transaction ID	Shipping Addre	Item ID	Reference	Quantity
2	04/05/2021		Express Checkout Payment	Completed	GBP	20	-0.88	19.12			2R9035 384E	David, Norman	bpi173685of		2
3	04/05/2021		Shopping Cart Item	Completed	GBP	20					2R9035 384E		bpi173685of		1
4	04/05/2021		Shopping Cart Item	Completed	GBP	0					2R9035 384E				1
5	04/05/2021		Express Checkout Payment	Completed	GBP	16.85	-0.79	16.06			012641 11V	Tammy Legget	ba150quof, ma100quof		3
6	04/05/2021		Shopping Cart Item	Completed	GBP	6.85					012641 11V		ba150quof		1
7	04/05/2021		Shopping Cart Item	Completed	GBP	10					012641 11V		ma100quof		1

Paypal Raw Data in CSV File

The Paypal online portal allows users to export account data in CSV format. The above image shows and example of this.

1. Customer name (blurred for privacy reasons).
2. Line-item Type.
3. Gross value in GBP of the sales item.
4. Paypal's own Transaction ID. This is imported into the Orixa App to ensure it is possible to trace the order back to its Paypal data if needed.
5. Email details for the customer (blurred for privacy reasons)
6. Item ID, this is the unique code used in the Orixa App to identify the products that have been ordered.

SQL of the "PayPalImport" procedure

```
CREATE PROCEDURE "PayPalImport" (IN "aFileName" VARCHAR(200) COLLATE "ANSI")
BEGIN
DECLARE Crsr CURSOR FOR Stmt;
DECLARE ErrMessage VARCHAR;
PREPARE Stmt FROM
' SELECT * FROM Information.TemporaryTables
WHERE Name = ''ImportOnlineSales'' ';
OPEN Crsr;
IF ROWCOUNT(Crsr) > 0 THEN
EXECUTE IMMEDIATE
' DROP TABLE "ImportOnlineSales" ';
END IF;

EXECUTE IMMEDIATE
```

```
' CREATE TEMPORARY TABLE "ImportOnlineSales"
("Date" DATE,
"Time" VARCHAR(20),
"Time zone" VARCHAR(20),
"Name" VARCHAR(60),
"Type" VARCHAR(50),
"Status" VARCHAR(50),
"Currency" VARCHAR(50),
"Gross" VARCHAR(20),
"Fee" VARCHAR(20),
"Net" VARCHAR(20),
"From Email Address" VARCHAR(120),
"To Email Address" VARCHAR(120),
"Transaction ID" VARCHAR(50),
"Shipping Address" CLOB,
"Item ID" VARCHAR(500),
"Reference TNX ID" VARCHAR(500),
"Quantity" VARCHAR(20),
"Receipt ID" VARCHAR(40),
"Contact Phone Number" VARCHAR(40)
) ';

EXECUTE IMMEDIATE
' SET Files Store TO "Imports"';

PREPARE Stmt FROM
' SELECT * FROM Configuration.Files
WHERE Name = ''' + aFileName + ''' ';
OPEN Crsr;
IF ROWCOUNT(Crsr) > 0 THEN
SET ErrMessage = 'Unable to import data, file-name "' + aFileName + '" may not be found.' + #13 +
'Please confirm this file-name is correct. It should be a simple name (no "path")
and the file must be present in the "Imports" store on the server';
EXECUTE IMMEDIATE
' IMPORT TABLE "ImportOnlineSales"
FROM "' + aFileName + '"
IN STORE "Imports"
FORMAT DELIMITED
DATE FORMAT ''DD/MM/YYYY''
USE HEADERS
';
EXECUTE IMMEDIATE
' UPDATE ImportOnlineSales
SET "Item ID" = ''DEL10''
WHERE (("Item ID" IS NULL) OR ("Item ID" = '''')) 
AND Gross <> ''0'' ';
SET ErrMessage = 'Unable to ALTER "ImportOnlineSales" data-table.';
EXECUTE IMMEDIATE
' ALTER TABLE ImportOnlineSales
ADD COLUMN ProductsID INTEGER ';
SET ErrMessage = 'Unable to UPDATE "ImportOnlineSales" data-table. Stage 1 failed.';
EXECUTE IMMEDIATE
' UPDATE ImportOnlineSales
SET
"Gross" = REPLACE('','' WITH '''' IN "Gross"),
"Fee" = REPLACE('','' WITH '''' IN "Fee"),
"Net" = REPLACE('','' WITH '''' IN "Net"),
"Shipping Address" = REPLACE('', '' WITH '','' + #13 IN "Shipping Address"),
"Name" = "Name" + COALESCE(#13 + "Contact Phone Number", '''');';
SET ErrMessage = 'Unable to UPDATE "ImportOnlineSales" data-table. Stage 2 failed.';
EXECUTE IMMEDIATE
' UPDATE ImportOnlineSales IOS
SET ProductsID =
```

```
(SELECT ID
FROM Products
WHERE ProdCode = IOS."Item ID") ';
SET ErrMessage = 'Unable to UPDATE "ImportOnlineSales" data-table. Stage 3 failed.';
EXECUTE IMMEDIATE
' UPDATE ImportOnlineSales IOS
SET Quantity = Quantity / (SELECT CountUnits
FROM Products
WHERE ProdCode = IOS."Item ID") ';
END;
```

The above procedure generates an "ImportOnlineSales data-table, which is well formed and includes all needed data-fields to link into the Orixa App.

A second procedure works through the "ImportOnlineSales" data-table, row by row and INSERTS the records, as is done in the "Valuations" example above.