Beginning SQL Reporting

Syllabus

This course will introduce learners to reporting with SQL, and some applications that use it in the SAP Business One ecosystem. You may be writing a report, measuring a KPI or setting up a dashboard by the time this course is over.

I have two goals for the course

- You are able to write your own SQL reports for use in reporting, KPIs and dashboards
- You can better communicate what report, dashboard, KPI or form you are looking for IT or our external consultants to make for you.

This is a multiweek course. We'll be covering a topic, then moving on the next week. There will be exercises for you to complete between classes.

Each week, except the first week, we will have half an hour of instruction, and then time afterwards to go over the homework.

Course Schedule

2/17/21 Introduction and Tools

- Identify at least five master tables by table name
- Define table, column and row.
- Find a table and column for most data
- Find the Query Generator
- Write a simple customer list using the generator
- Define SELECT , FROM and WHERE SQL keywords
- Save a query to their sandbox
- Run a query from the query manager
- Explain the SDL protocol for Saving SQL
- Export your report to Excel

2/24/21 Selection and Functions

- Edit SQL statements directly
- Identify Types of Columns
- Use the logical operators AND, OR and NOT
- Create reports using multiple Criteria
- Use the CASE expression
- Use the ORDER BY expression
- Use the DESC operator

3/3/21 Date Functions and Parameters

- Rename columns with AS
- Use a date in a WHERE Clause
- Use the GETDATE function
- Use the DATEADD function
- Use DAY, MONTH, YEAR
- Use the DATEDIFF Function

3/10/21 Debugging

- Identify the three common bug types
- Correct syntax errors
- Correct logical Errors
- Prevent or handle data errors
- Use CAST
- Identify Null values with ISNULL
- Use ISDATE, ISNUMBER
- Add Comments

3/17/21 JOINS

- Define one to one and one to many relationships
- Identify Child tables
- Identify keys and related tables
- JOIN a Child table to a Parent Table
- JOIN a key to its related tables

3/24/21 More Joins

- LEFT JOIN
- Inner and Outer joins
- UNION
- Subqueries

3/31/21 Grouping and Aggregate functions

- Learn the functions COUNT, SUM,AVG
- Learn the GROUP BY Clause

4/7/21 UDF's and variables

- Read UDF descriptions
- Use variables

4/14/21 – 4/28/21 Dashboards

Objectives to be determined

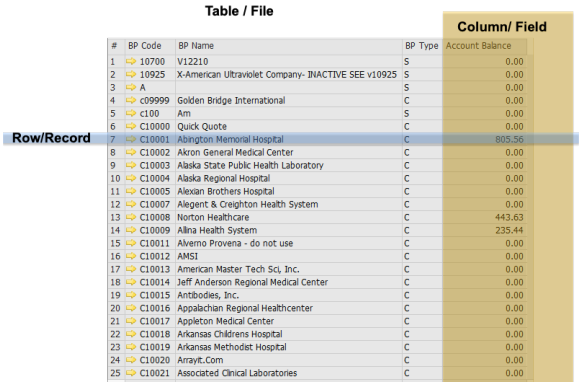
Part 1 – Introduction

- Identify at least five critical tables by name
- Define table, column and row.
- Find a table and column for most data
- Find the Query Generator
- Write a simple customer list using the generator
- Define SELECT , FROM and WHERE SQL keywords
- Save a query to their sandbox
- Run A query from the query manager
- Explain the SDL protocol for Saving SQL
- Export your report to Excel

Where you Find SQL:

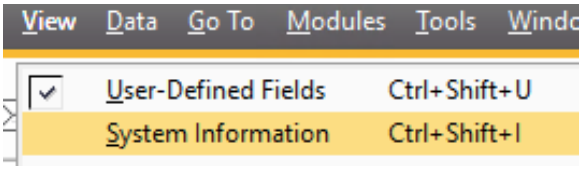
- Query manager for reports
- User Defined values(🔎)
- Crystal Reports
- Boyum B1 UP Validations
- Boyum B1 UP Macros
- Dashboards
- Data for mass changes(DTW)

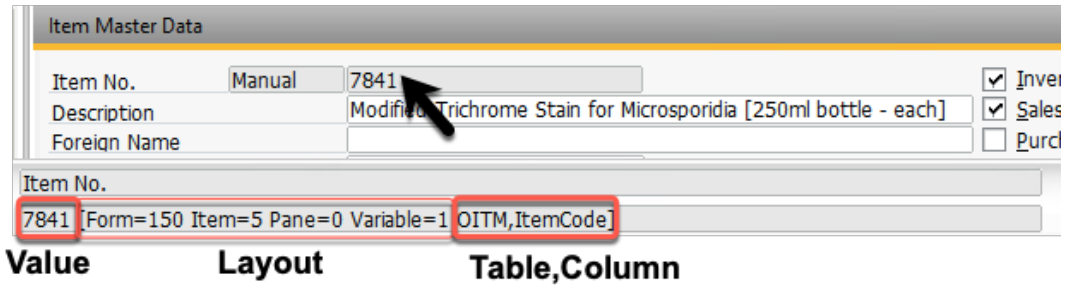
Definitions



#	BP Code	BP Name	BP Type	Account Balance
1	10700	V12210	S	0.00
2	10925	X-American Ultraviolet Company- INACTIVE SEE v10925	S	0.00
3	A		S	0.00
4	c09999	Golden Bridge International	C	0.00
5	c100	Am	S	0.00
6	C10000	Quick Quote	C	0.00
7	C10001	Abington Memorial Hospital	C	805.56
8	C10002	Akron General Medical Center	C	0.00
9	C10003	Alaska State Public Health Laboratory	C	0.00
10	C10004	Alaska Regional Hospital	C	0.00
11	C10005	Alexian Brothers Hospital	C	0.00
12	C10007	Alegent & Creighton Health System	C	0.00
13	C10008	Norton Healthcare	C	443.63
14	C10009	Allina Health System	C	235.44
15	C10011	Alverno Provena - do not use	C	0.00
16	C10012	AMSI	C	0.00
17	C10013	American Master Tech Sci, Inc.	C	0.00
18	C10014	Jeff Anderson Regional Medical Center	C	0.00
19	C10015	Antibodies, Inc.	C	0.00
20	C10016	Appalachian Regional Healthcenter	C	0.00
21	C10017	Appleton Medical Center	C	0.00
22	C10018	Arkansas Childrens Hospital	C	0.00
23	C10019	Arkansas Methodist Hospital	C	0.00
24	C10020	Arrayit.Com	C	0.00
25	C10021	Associated Clinical Laboratories	C	0.00

Table and Column Names





O + 3 Letters = Master/Parent table
Example: ORDR – Sales Order Header

3 Letters + Number = Sub/Child Table
Example: RDR1 – Sales Order Rows

Note: Until Mid-March we will be working primarily with Master tables and a few child tables, including RDR1 and WOR1. To join the parent and child is one of the hardest concepts in SQL, so we are saving it for later.

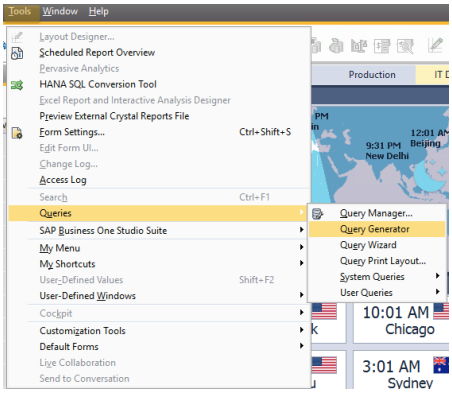
Common Master Table Names

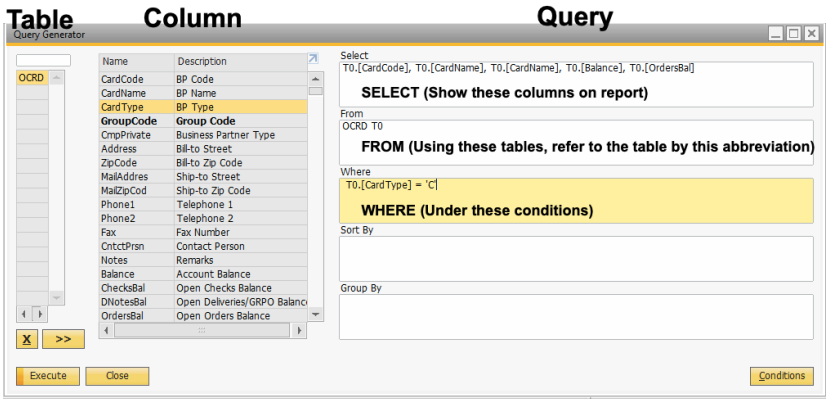
ID	Module	Table	Remember	Description	Common Columns Used (Key in Bold, Linked table in {})
	Business Partners				
2	Business Partner Master	OCRD	business **CaRD**.	Customer and Vendor master data	**CardCode**, CardName,CardType, Address,ZipCode, City Country, Balance
	Marketing Docs (A/R) & (A/P)				
17	Sales Orders	ORDR	sales o**RDeR**	Information about a specific sales order; find items ordered in RDR1; use if tracking sales	**DocEntry**, CardCode{OCRD},CardName, Address, DocDate, DocDueDate, NumAtCard, DocTotal, OwnerCode, GrosProfit, TrackNo, DocStatus
13	A/R Invoices	OINV	sales **INV**oice	Information about a specific A/R Invoice; find items ordered in INV1; use if tracking money to collect	**DocEntry**, CardCode{OCRD},CardName, Address, DocDate, DocDueDate, NumAtCard, DocTotal, OwnerCode, GrosProfit, TrackNo, DocStatus
22	Purchase Order	OPOR	**P**urchase **OR**der	Information about a specific purchase order; find items ordered in POR1	**DocEntry**, CardCode{OCRD},CardName, Address, DocDate, DocDueDate, NumAtCard, DocTotal, OwnerCode, DocStatus
18	A/P Invoices	OPCH	**P**ur**CH**ase goods	Information about a specific A/P Invoice; find items ordered in PCH1	**DocEntry**, CardCode{OCRD},CardName, Address, DocDate, DocDueDate, NumAtCard, DocTotal, OwnerCode, DocStatus
	Inventory/Production				

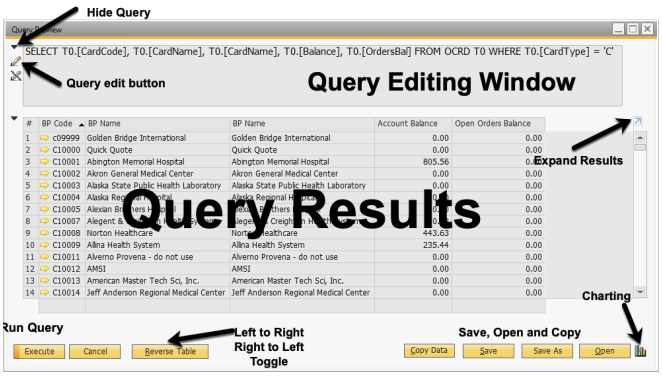
4	Inventory Item	OITM	inventory **ITeM**	Individual Inventory items, including Raw materials, labor, fees, and finished goods	**ItemCode**, ItemName, OnHand, IsComitted, OnOrder, AvgPrice
66	Bill of Materials	OITT	**I**nven**T**ory **T**ree	Combination of inventory items to make a subassembly or finished good which items are in BOM are in ITT1	**Code,**Qauntity,PriceList
202	Production Order	OWOR	**WOR**k order		**DocEntry,** ItemCode{OITT},Status, PlannedQty, CmpltQty, RjctQty, PostDate, DueDate, CloseDate, RlsDate

Writing Queries

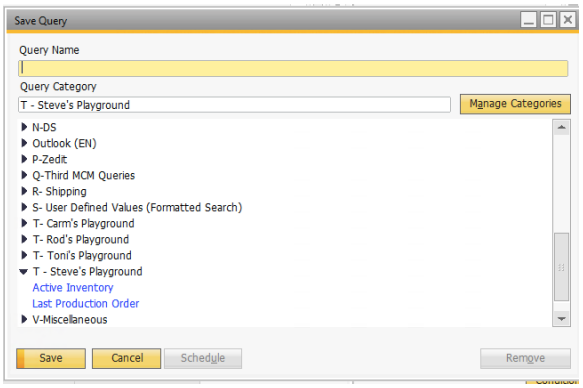
The Query Generator







Saving



SDL Protocol for SQL Saving and Production Report Release

Have a playground to save your work to. If you are in this course, your playground has already been created. You can do anything you want in your playground.

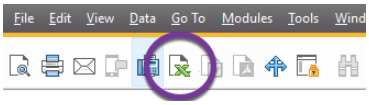
If you have a report for general distribution to SDL, submit a request to IT for the Query you'd like to make part of the system. The submitted query must have in the comments the following information

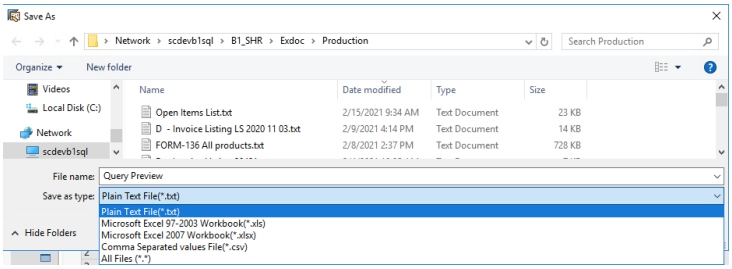
- The name of the report
- What it does
- When you wrote it
- Your name
- What category to place it in Query Manager

Once submitted, IT will test it. If it works to IT's satisfaction, will place it in the desired category for general use, and notify users.

Approval will also be required for queries used in other places such as Dashboards.

Excel Exporting





Assignment 1:

Make one of the following reports. You choose what columns you want, but include the column to get the golden arrow to the full record.

1. An Open (O) sales orders report
2. An Open (O) invoices report
3. A WIP report of all released (R) production orders

Save it to your sandbox.

Then run the report again and export it to an Excel spreadsheet.

Extra credit: Change report 1 to report 2 above with a single change.

Part 2: Selection and Functions

- Edit SQL statements directly
- Identify Types of Columns
- Use the comparison operators =, >, <, >=, <=, <>
- Use the logical operators AND, OR and NOT
- Create reports using multiple Criteria
- Use the CASE expression
- Use the ORDER BY expression
- Use the DESC operator

3 Major Data Types

- Number - A numerical value, can use mathematical (+,-,*,/) operators to get a different result. Often used for Row ID's like `DocEntry` and `DocNum`, Quantities and Price values.
- VarChar(String) – A series of characters which make up text. Names, addresses, and comments are strings. Some ID's like `ORDR.Cardcode` are also strings. We express strings with Apostrophes like this `'This is a string'`
- Date – a date and time stored together. Posting, completion, and delivery dates are dates. We express dates as though they are a string `'2/12/21'`
- Boolean – true or false , Yes or no values. Often in SAP B1 a single VarChar will be used instead such as . Checkboxes are often Boolean values.

Boolean Expressions

Boolean expression take two values and give a true or false value to explain their relationship. For example
`1 = 2` is false `1 = 1` is true
You can use more that equals though. There is

Greater than **(>)**	`1 > 2` is false,	`1 > 1` is false
Less than **(<)**	`1 < 2` is true,	`1 < 1` is false
Not Equal **(<>)**	`1<>2` is true,	`1<>1` is false

And there are the combinations of

Greater than or Equal **(>=)**	`1 >= 2` is false,	`1 >= 1` is true
Less than or Equal **(<=)**	`1 =< 2` is true,	`1 =< 1` is true

This works for all types. VarChars will compare to alphabetical order and are case sensitive, and Dates to time order.

`'01/01/21' = '02/01/21'` is False `'01/01/21' < '02/01/21'` is True
`'Steve' <> 'steven'` is True
`'BD' < 'Fisher'` is True

Complex Boolean Expressions

There are three more operator you can use `AND` `OR` and `NOT`.

AND takes two Boolean expressions and if they are both true is true:

False	AND	False	False
False	AND	True	False
True	AND	False	False
True	AND	True	False

OR takes two Boolean expressions and if at leat one of them is true the result is true.

False	OR	False	False
False	OR	True	True
True	OR	False	True
True	OR	True	True

NOT flips the value, and goes in front of the value

NOT	true	false
NOT	false	true

Multiple selection with WHERE

We can use all of this to make more sophisticated selections. For example, Suppose I have this Query of Delvery documents:

```
SELECT T0.[DocEntry], T0.[DocStatus], T0.[DocDate],
T0.[CardCode], T0.[CardName], T0.[NumAtCard], T0.[DocTotal] FROM
ODLN T0
```

I want only the ones from February 2021. I can do this:

```
SELECT T0.[DocEntry], T0.[DocStatus], T0.[DocDate],
T0.[CardCode], T0.[CardName], T0.[NumAtCard], T0.[DocTotal] FROM
ODLN T0 WHERE T0.DocDate >= '2/1/2021' AND t0.docDate <
'3/1/2021'
```

Anothere example

```
SELECT T0.[CardCode], T0.[CardName], T0.[Balance], T0.OrdersBal
FROM OCRD T0 WHERE T0.[CardType] = 'C'
```

This gets us customer order and invoice balances. But suppose we wanted to get customer balances for international customers only. I find two true conditions and connect them with AND

```
SELECT T0.[CardCode], T0.[CardName], T0.[Balance], T0.OrdersBal
FROM OCRD T0 WHERE T0.[CardType] = 'C' AND NOT  T0.Country =
'US'
```

I look for all US customers first. And then I'll look for all non-US customers.

Functions

I can go one more step. I'll look for all customers that do not use family or standard pricing. This requires a function. A function is a keyword with a Parentheses and several values called parameters within the parenthesis in them.

One we use with Boolean expressions is `IN().` it has a syntax of
`<Value> IN(<Value1>,…<ValueX>)`

For example,
`T0.ListNum IN(1,2,7,11,14)`

Is true when you have a item cost, standard, or family pricing list. I can use not

```
NOT  T0.ListNum IN(1,2,7,11,14)
```

To find everyone else. I can add that to the full report and get all international customers on a distributor price list.

```
SELECT T0.[CardCode], T0.[CardName], T0.[Balance], T0.OrdersBal
FROM OCRD T0 WHERE T0.[CardType] = 'C' AND NOT  T0.Country =
'US' AND NOT T0.ListNum IN(1,2,7,11,14)
```

AND there is our list.

String Comparisons with LIKE

When working with Strings, a common way to find data is with `LIKE`. Like's syntax is
`LIKE(<String Value>)`
What makes LIKE so powerful is it uses the wildcard %. Wild cards mean anything is true.
For example
`LIKE('S%')` would be true for all items beginning with *S*.
`LIKE('%FIXED')` would be true all items ending with *FIXED*
`LIKE ('%Slide%')` would be true for all items that contain *Slide* in the string.

This Query

```
SELECT T0.ItemCode, T0.ItemName FROM OITM T0
WHERE T0.ItemName = 'Shrink Seals (92X25mm) [400 seals/pkg] '
```

Change it to this

```
SELECT T0.ItemCode, T0.ItemName FROM OITM T0 WHERE T0.ItemName
Like( 'Shrink Seals%')
```

Finds all descriptions beginning with Shrink seals.

```
SELECT T0.ItemCode, T0.ItemName FROM OITM T0 WHERE T0.Dscription
LIKE('X-%')
```

Does find inactive items in inventory. I prefer to inactivate the item and use the column validFor.

```
SELECT T0.ItemCode, T0.ItemName FROM OITM T0 WHERE t0.ValidFor =
'N'
```

However neither covers all cases in our data so you may want to use both.

```
 SELECT T0.ItemCode, T0.ItemName FROM OITM T0 WHERE T0.ItemName
LIKE('X-%') OR t0.ValidFor = 'N'
```

Sorting

I can sort the International Distributor report in the Query by ORDER BY. You can do this from the Query Generator or editing the Query. I tend to edit the query. I'll sort these by their billing amounts and then by their Sales balance.

```
SELECT T0.[CardCode], T0.[CardName], T0.[Balance], T0.OrdersBal
FROM OCRD T0 WHERE T0.[CardType] = 'C' AND NOT  T0.Country =
'US' AND NOT  T0.ListNum IN(1,2,7,11,14) ORDER BY T0.[Balance],
T0.OrdersBal
```

And executing that report gives us a great summary. However, I might want it with the biggest number first. I'll Use DESC for *descending*.

```
SELECT T0.[CardCode], T0.[CardName], T0.[Balance], T0.OrdersBal
FROM OCRD T0 WHERE T0.[CardType] = 'C' AND NOT  T0.Country =
'US' AND NOT  T0.ListNum IN(1,2,7,11,14) ORDER BY T0.[Balance]
DESC, T0.OrdersBal DESC
```

CASE

Let write a report for Inventory items and their work centers, based on a User defined field of U_Workctr.

```
SELECT T0.[ItemCode], T0.[ItemName], T0.[U_Workctr],
T0.[OnHand], T0.[IsCommited], T0.[OnOrder] FROM OITM T0 WHERE
T0.[IsCommited] > T0.[OnHand] And  T0.[SellItem] = 'Y'
```

Here is a table of the values for U_Workctr

A	High Speed Slide Printing
B	R&D
M	Manual
S	Slide Printing & Coating
L	Laboratory
H	Fabrication/Machine Shop
E	Electronics
O	Others
F	Filling
C	N/A

To make the report more readable for those who don't know the codes, I can use CASE() to change the output.

The general form is

```
CASE <Column>WHEN <value> THEN <newValue>…ELSE <defaultValue>
END
```

For example I can do this to use the words slides and filling. :

```
CASE T0.[U_Workctr] WHEN  'S'  THEN 'Slides' WHEN 'F' THEN
'Filling' ELSE t0.U_WorkCtr END
```

SO my Query now looks like this.

```
SELECT T0.[ItemCode], T0.[ItemName],CASE  T0.[U_Workctr] WHEN
'S'  THEN 'Slides' WHEN 'F' THEN 'Filling' ELSE t0.U_WorkCtr
END, T0.[OnHand], T0.[IsCommited], T0.[OnOrder] FROM OITM T0
WHERE T0.[IsCommited] > T0.[OnHand] And  T0.[SellItem] = 'Y'
```

Assignment:

Pick one:

Write a Report using RDR1 that reports the quantity and line item total for all Shrink Seals between 2/1/2020 and 2/1/2021. Show the highest quantity first. Indicate if the item has been closed or open with the words **Closed** or **Open**. To help you out, here's a couple of useful columns.

Field	Description	SQL Type	Constraints
DocEntry	Document Internal ID	int	
LineNum	Row Number	int	

LineStatus	Row Status	char	C=Closed, O=Open
ItemCode	Item No.	nvarchar	
Dscription	Item/Service Description	nvarchar	
Quantity	Quantity	num	
ShipDate	Row Delivery Date	date	
OpenQty	Remaining Open Quantity	num	
Price	Unit Price	num	
Currency	Price Currency	nvarchar	
Rate	Currency Rate	num	
DiscPrcnt	Discount % per Row	num	
LineTotal	Row Total	num	

Write a report Using OWOR for Production orders for all of February 2021 that has a work center of Slide printing or High speed slide printing. Have a column (`U_dept`) stating **Slides/Coating** or **High Speed** , the item number, and the planned Quantity and the status of the order. Put the highest Planned Quantity first on the table.

Field	Description	SQL Type	Constraints
DocEntry	Internal Number	int	
DocNum	Document Number	int	
Series	Series	int	
ItemCode	Product No.	nvarchar	
Status	Production Order Status	char	C=Canceled, L=Closed, P=Planned, R=Released
Type	Production Order Type	char	D=Disassembly, P=Special, S=Standard
PlannedQty	Planned Quantity - Header	num	
CmpltQty	Completed Quantity	num	
RjctQty	Rejected Quantity	num	
PostDate	Order Date	date	
DueDate	Due Date	date	
OriginAbs	Production Order Origin Entry	int	

SCHEDULING NOTE:

I will be out of the office next week recording *SAP Business One: Logistics and Production*. There will not be class. You have two weeks to finish your assignments. I will make two videos

ready for everyone next week. The first will be the material covered in class on the WHERE clause. We didn't get to the CASE I included above, and I will make a second video to discuss that, which you will need for your assignment. I suggest getting up to the part you need the CASE (closed/Open) (high speed/filling) by next Wednesday so you only have that to work on the following week.

Review

Write a Report using RDR1 that reports the quantity and line item total for all Shrink Seals between 2/1/2020 and 2/1/2021. Show the highest quantity first. Indicate if the item has been closed or open with the words **Closed** or **Open**.

```
SELECT
T0.ItemCode,T0.Dscription,
     T0.Quantity,T0.LineTotal,
     CASE T0.LineStatus
          WHEN 'C' THEN 'Closed'
          ELSE 'Open'
     END AS "Order Status"
FROM RDR1 T0
WHERE
T0.DocDate >= '2/1/2020'
AND T0.DocDate <= '2/1/2021'
AND T0.Dscription LIKE('%Shrink%')
ORDER BY
     T0.Quantity DESC
```

Write a report Using OWOR for Production orders for all of February 2021 that has a work center of Slide printing or High speed slide printing. Have a column (`U_dept`) stating **Slides/Coating** or **High Speed** , the item number, and the planned Quantity and the status of the order. Put the highest Planned Quantity first on the table.

```
SELECT
T0.DocNum, T0.ItemCode,T0.PlannedQty AS "Planned Qty",
CASE T0.U_dept
   WHEN 'A' THEN 'High Speed'
   WHEN 'S' THEN 'Slides/Coating'
   ELSE 'Non-Slide Department'
END as "Slide Line",
CASE T0.Status
   WHEN 'P' THEN 'Planned'
   WHEN 'R' THEN 'Released'
   WHEN 'L' THEN 'Closed'
   ELSE 'Canceled'
END as "Work Order Status"
FROM OWOR T0
WHERE
   T0.CreateDate >= '2/1/21' AND
   T0.CreateDate < '3/1/21' AND
   T0.U_dept IN('A','S')
ORDER BY T0.PlannedQty DESC
```

Date Functions and Parameters

- Parameters
- Use the GETDATE function
- Use the DATEADD function
- Use DAY, MONTH, YEAR
- Use the DATEDIFF Function

Parameters

In the last exercise we had this selection criteria

```
WHERE
   T0.DocDate >= '2/1/2020'
   AND T0.DocDate <= '2/1/2021'
   AND T0.Dscription LIKE('%Shrink%')
```

This contains literal values for dates. That means you change your code every time you run it if you want different dates. We can ask the user for input using parameters. If we change the code to this:

```
WHERE
   T0.DocDate >= [%0]
   AND T0.DocDate <= [%1]
   AND T0.Dscription LIKE('%Shrink%')
```

When we run the code, the system will ask us for those dates.
A parameter is formed by square brackets, a % sign and a number between 0 and 19. You'll rarely use more than two.
SAP B1 replaces the parameter code with the value added in the selection criteria *before* the code runs. You can use parameters for any value, but you need to be careful. For example, add this line

```
AND T0.LineStatus = '[%2]'
```

Notice I put quotes around the parameter. The parameter will be an O or a C. For SQL to recognize it use quotes to indicate a varchar. Run this and you can select open or closed orders for shrink seals.

GETDATE()

While parameters work in some situation where we are okay specifying dates, date function will automatically calculate dates for us. This is especially useful for dashboards. Functions are values found for us, which we sometimes pass arguments, values that determine a final result. A very simple function **GETDATE()** gives today's date without any parameters. For example, change our code to

```
AND T0.DocDate <= GETDATE()
```

This changes our query to pick a date before today and find all cases up to and including today.

MONTH(), DAY(),YEAR()

Other functions can give you more date information. For example, suppose I want to know only the month of a date. I can use the `MONTH()` function,

```
T0.DocDate,MONTH(T0.DocDate), T0.ItemCode,T0.Dscription,
```

Run and we can see the date. I can use this to find data in specific months.

```
MONTH(T0.DocDate) = 3
```

I can also do this to years. to find last calendar year's data in any case but January:

```
MONTH(T0.DocDate) = MONTH(GETDATE())
AND YEAR(T0.DocDate) = YEAR(GETDATE()) - 1
```

You can also do this to days as well with `DAY()`.

DATEADD()

That expression is a little difficult to use for previous months. For true date calculations we use `DATEADD`. `DATEADD` has three parts

1. The date unit you'll be adding
2. The value you'll be adding
3. The date you'll be adding it to.

In the date part you can use these units to compute a new date.

Unit	Long form	Short form
Year	`year`	`yy or yyyy`
Quarter	`quarter`	`qq or q`
Month	`month`	`mm or m`
Day	`day`	`dd or d`
Week	`week`	`ww or wk`

The second part is the amount you are adding. If you want to go back in time, you use a negative number.

The last is the date you are adding to, this usually will be a GETDATE() or a column.

For example

```
DATEADD(dd,14,GETDATE())
```

is 14 days from today

```
DATEADD(year,-1,GETDATE())
```

Is a year ago

```
DATEADD(m,6,GETDATE())
```

Is six months from now.

DATEDIFF()

DATEDIFF is the opposite of DATEADD. It tells you the interval between two dates in the unit you specify. For arguments it will have

1. The units you will measure by. Use the same units as DATEADD
2. One date to subtract from
3. The other date to subtract from

In the example production order report, we have

```
DATEDIFF(day,T0.DueDate,T0.CloseDate) AS "Closing Difference" ,
DATEDIFF(day,T0.CreateDate,T0.CloseDate) AS "Days Open",
DATEDIFF(day,GETDATE(),T0.DueDate) AS "Past Due"
```

Exercises

Change the production order report to give all production orders from a day in the past to a day three weeks from now. Allow your user to select any one department.

Make a report from any table of your choice that lists two dates and the number of days between them. Add any extra columns necessary for this report to make sense. This report should have a from one of the two dates you used a range from today to a month ago.

Using Two Tables: Joins

Table Relationships

For some reports one table is enough. In many cases you will need more than one table. The JOINs link tables together. There are two kinds of relationships. Suppose you had this report:

#	Document Number	Product No.	WOR Status	Closing Difference	Days Open	Past Due	Work Center	User Signature	Customer Code
1	21464	6030	Closed			-20	F	19	
2	21465	261	Closed			-20		21	
3	21466	41-LSM-0406-BK	Canceled			2	S	19	C12689
4	21467	74-AXI-1000-PL-BK	Canceled			2	S	19	C12752
5	21468	041-MOLDEV24	Canceled			-17	S	19	C12739
6	21469	041-MOLDEV64	Canceled			-17	S	19	C12739
7	21470	050-2S-100	Closed	-1		-19	M	19	C12153

The **User Signature** and **Customer Code** are codes. It would be nice if we had descriptions of them. However, the descriptions are not in the OWOR table used for this report, but in two different tables OUSR and OCRD.

A JOIN lets us link tables in such a way that you can get information from the one record associated with the user signature or the Customer code. There is one customer code associated with one record. We call this a **one-to-one relationship**.

If I were to look at one of these production orders, I'd find a table of components inside of them.

Type	Standard	No. Primary	21468
Status	Canceled	Order Date	03/09/21
Product No.	[redacted]	Start Date	03/09/21
Product Description	[redacted]	Due Date	03/12/21
Planned Quantity	125 UoM Name EA	User	Dale Patton
Warehouse	FG	Origin	Manual
Priority	100	Sales Order	55466
Routing Date Calculation	On Start Date Update Now	Customer	C12739
		Distr. Rule	
		Project	

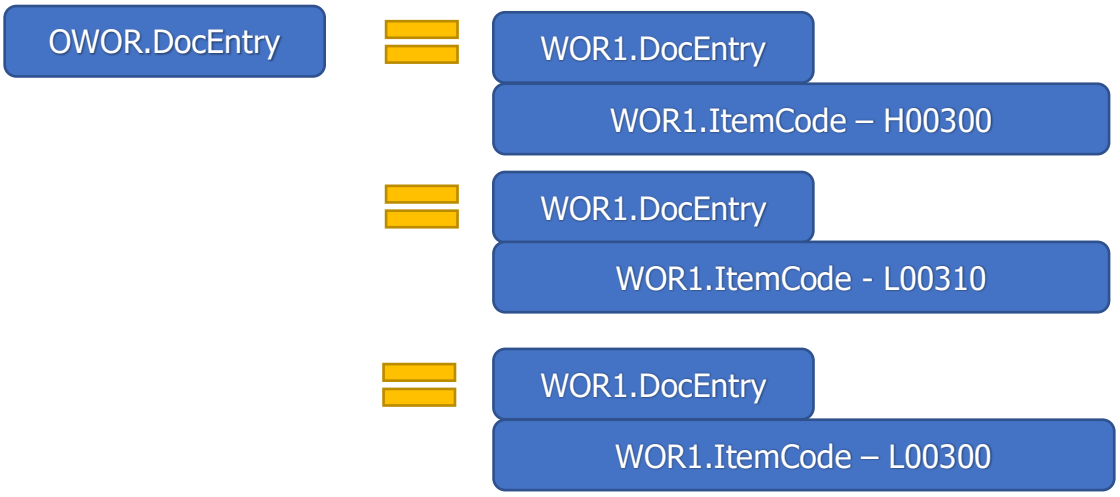
Components | Summary

Type	No.	Description	Base Qty	Base ...	Planned Qty	UoM N...	Available	In Stock	Ordered	Start Date	Committed	End Date	UoM ...	Route ...
Item	H00300	Overhead - Slide Printing &		1/50000	0.003	MIN	0			03/09/21		03/12/21	Manual	0
Item	L00310	Labor FIXED - Slide Printing	0.48	12/25	60	MIN	0			03/09/21		03/12/21	Manual	0
Item	L00300	Labor - Slide Printing & Coa	1.8	9/5	225	MIN	0			03/09/21		03/12/21	Manual	0
Item	RM0843	Ink Black Nazdar [1000g/car	0.34	17/50	42.5	G	2,223.017	2,453.405		03/09/21	230.388	03/12/21	Manual	0
Item	RM0922	Teflon (Zonyl) [44.1lb/drum] (	0.26	13/50	32.5	G	14,335.965	4,446.558		03/09/21	110.593	03/12/21	Manual	0

Each item in the items table is a separate item in another table, WOR1. We match data from the Header in OWOR with each row in WOR1. There is only one Header, but can be many rows. This is called a **one-to-many relationship.**

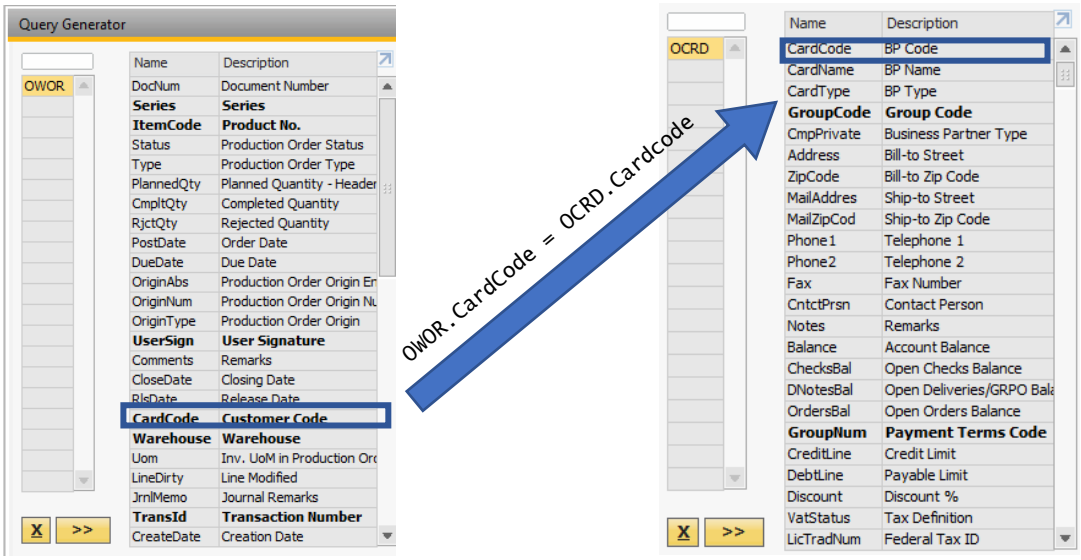
Let's look at this graphically. In a One-to-one relationship, the BP number in the Production order connects to the BP with the same number, which give us access to the Business partner table OCRD.

I can then use `CardName` in my report to print the name of the Customer on the report.
In a one-to-many relationship, there is a column, often called `DocEntry`, on the rows, each has different data, such as an `ItemCode`.



Links and Keys

IN order to make a relationship, we need a column on one table that has a matching value on another table. From the examples above I have a column `CardCode` in both the `OWOR` and `OCRD` tables. In a one-to-one relationship, when the value in the BP Master data matches the one in the production order, then we can use the rest of the data in the Business partner for that customer.



In the query generator, the bold columns listed are what are called *links*. They point to a column in another table that you can join together.

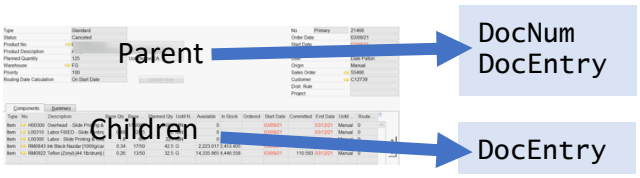
The column in the other table is called a *key*. Keys are fundamental to any database with more than one table. They are what you use to link the tables together. Keys have some very specific attributes:

- **They are immutable**. You cannot change the value of a key once you make it.
- **They are unique**. You only have one of this value in any database
- **They are simple datatypes**. For speed in searching and to keep them unique, keys are usually numbers. However, you will find strings for keys as well, such as our case of `CardCode`. Dates don't work well for keys, as its likely to have multiple of the same date.

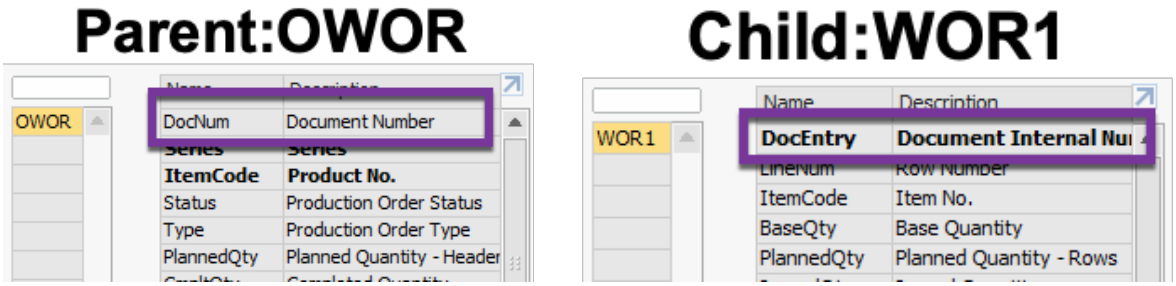
You *never* define keys. SAP does that for you.

DocNum and DocEntry Explained

In one-to-many relationships, you'll often see a relationship called a Parent-Child relationship. It means that then many items are dependent on the one item. Marketing documents, BoMs and Production Orders are all examples of A parent-child relationship.



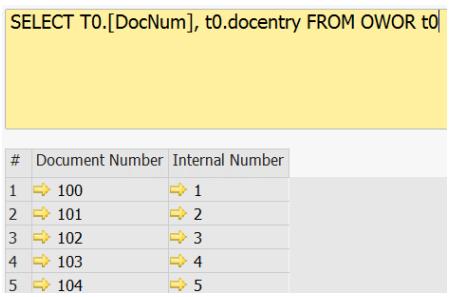
In parent documents, if you look at them from the Query Generator, you'll see a `DocNum`, with a description of **Document Numbe**r. In Child Documents, you'll see a `DocEntry`, with a description of **Document Internal Number**

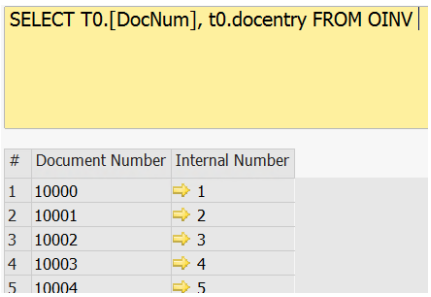


When looking at reports, you might find links between the Parent and child with `DocNum` and `DocEntry`.

This is usually okay. In most, but *not all* cases `DocNum` and `DocEntry` are the same. The true key in either a parent or child is `DocEntry`. Unfortunately, Query generator hides `DocEntry` on the parent table, so you won't find it easily.

Invoices and Production orders are example of the `DocEntry` not equal to the `DocNum`. Consider these queries:





When we initialized the system, we started at a different DocNum than 1 for these two. You'll get the wrong data if you link a DocNum and a DocEntry.

In general, link DocEntry to a key DocEntry, even if you don't see it

Golden Arrows

Also notice the difference between `DocEntry` and `DocNum`. `DocEntry` always has a golden arrow, `DocNum` might. Keys and implicit links will have golden arrows. SAP Business one places these automatic links without any work on your part.

When writing reports, remember who it is for and where the report will end up. If you are exporting a report to Excel, don't `SELECT DocEntry`. It will just confuse things. Use `DocNum` Instead. If you are leaving the report on SAP B1for interactive use, include `DocEntry` if the `DocNum` does not have a golden arrow.

INNER JOIN

Now that you understand these relationships and some of the bumps in the road, you're ready to start coding with them.

In SQL, we make these relationships with JOIN statement. There are several, but the most common is the `INNER JOIN`. You can use `INNER JOIN` two ways:

- In the Query generator adding a second table to your query automatically makes an inner join. Query generator is usually smart enough to make the right link. As mentioned earlier, this still could be wrong, so check your links.
- You can add it manually to the `FROM` Clause. Generally you add the highlighted part for the second table

```
FROM TABLE1 T0 INNER JOIN TABLE2 T1 ON T0.Key = T1.Link
```

Let's break this down with an example. I'll modify with this query listing production orders, so I can put the owner's name on it, instead of the number:

```
SELECT T0.[DocNum], T0.[ItemCode], T0.[UserSign], T0.[CardCode]
FROM OWOR T0
WHERE T0.[PostDate] >= DATEADD(day,-3,GETDATE())
```

I'll first concentrate on the FROM Clause, which already has one table, the production order header. I'll add the keywords INNER JOIN

```
FROM OWOR T0 INNER JOIN
```

The the name of the table I want to join to this one, which is OUSR. I'll also give it an abbreviation of T1.

```
FROM OWOR T0 INNER JOIN OUSR T1
```

Then add the ON keyword, and specify the link and key.I know the link is T0.[UserSign]. I've looked up the key T1.USERID. I make a test that they are equal.

```
FROM OWOR T0 INNER JOIN OUSR T1 ON T0.UserSign = T1.USERID
```

That joins the table together. Then I can use T1. To add information from that table, so I'll add the user's name to the SELECT Caluse using t1.

```
SELECT T0.[DocNum], T0.[ItemCode], T0.[UserSign],T1.U_NAME,
T0.[CardCode]
```

Run this and you'll see the names on the report.

Homework:

Explore links in SAP: Follow five golden arrows on the system. Write down the link and the key in the table below. Use system Information to find the field name.

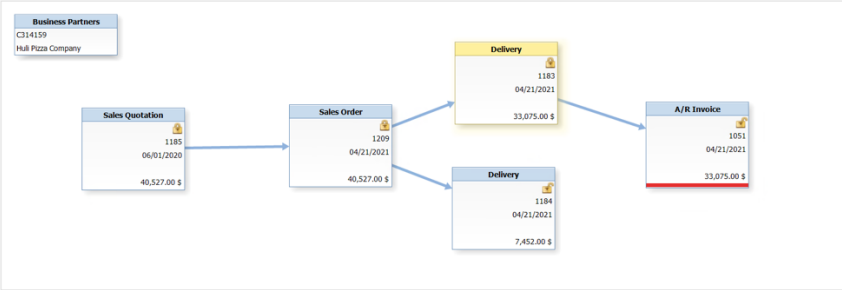
Link #	Module	Link		Key	
		Table	Field	Table	Field
1					
2					
3					
4					
5					

Change this report to include the name of the Business Partner. What changes between the current report and the modified one with the business partner name?

```
SELECT T0.DocNum, T0.ItemCode, T0.UserSign, T0.CardCode
FROM OWOR T0
WHERE T0.PostDate >= DATEADD(day,-3,GETDATE())
```

Joins and Relationship Maps

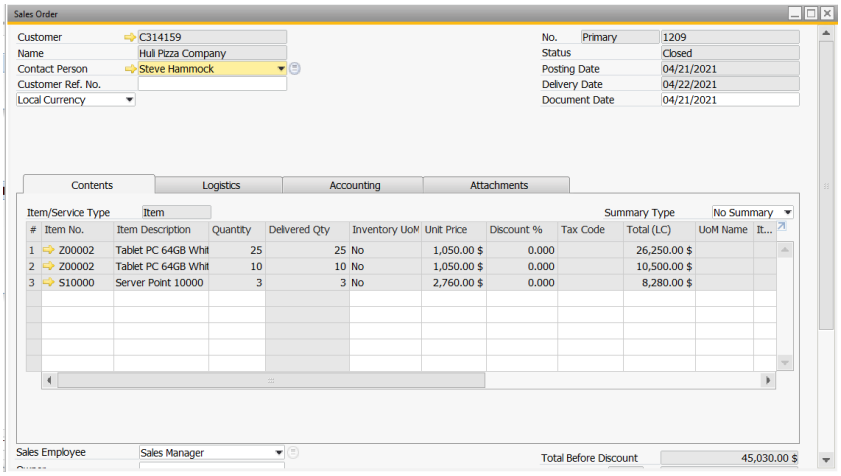
The relationship map is a powerful feature of SAP Business one linking different documents together. One of the most common uses is linking marketing documents together like this:



It would be nice to be able to link documents together like this with JOIN. You can do this, but there's a few tricks and techniques you'll have to know.

You's expect that you woulf link documents together, such as a field in ORDR for a sales order to one in ODLN for delivery. However, you can see the problem with that with the relationship map above. What if I had two delivery documents?

Here's the sales order for that:

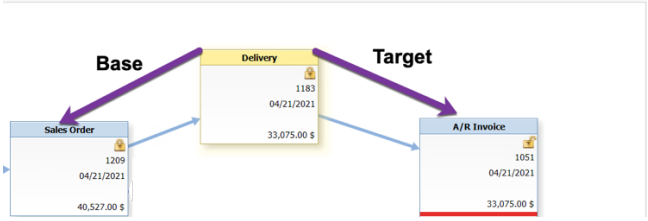


There is a partial delivery of the first two lines then a second delivery of the last line. What shows up on a given delivery is based on the rows of the document not the header.

This is why you find the links from one document to another not in the header but in the rows. You cannot guarantee what will show up where for a full document, but individul items an be tracked.

Targets and bases

How you link depends on several fields, depending on direction there are *Base* fields, which point to the past document and Target fields which point to future documents. If we look at a delivery document, it has a base of a Sales order, and a target of a A/R Invoice.



In a simple world we always have a base document that is a sales order here and a delivery that is always a A/R invoice. However, I could have a return on a delivery document. I could also get a delivery document directly from a quote. Just having a number to match to a docentry is not enough. You need to know where it points to.

For every link there are two parts: one is the docentry of the target or base, the other the type of the target and base.

Here's a table listing these:

Row Linking Fields for Relationship Maps

Field	Description	SQL Type	Length
DocEntry	Document Internal ID	int	11
LineNum	Row Number	int	11
TargetType	Target Document Type	int	11
TrgetEntry	Target Document Internal ID	int	11
BaseRef	Base Document Reference	nvarchar	16
BaseType	Base Document Type	int	11
BaseEntry	Base Document Internal ID	int	11
BaseLine	Base Row	int	11
LineStatus	Row Status	char	1

I can make a query like this to see them for the delivery documents

```
SELECT T0.[DocEntry], T0.[LineNum], T0.[ItemCode],
T0.[Dscription], T0.[TargetType], T0.[TrgetEntry],
T0.[BaseType], T0.[BaseEntry] FROM DLN1 T0
```

With a result of

#	Document Internal ID	Row Number	Item No.	Item/Service Description	Target Document Type	Target Document Internal ID	Base Document Type	Base Document Internal ID
390	1174		LMT-02B	Pizza Tablet Set - Batch	-1		17	1194
391	1175		Z-MRP1	Penguin Tablet	-1		17	1195
391	1176		Z-MRP1	Penguin Tablet	-1		17	1196
391	1177		Z-MRP1	Penguin Tablet	-1		17	1197
391	1178		Z-MRP1	Penguin Tablet	-1		17	1198
391	1179		Z-02	Penguin Tablet Set	-1		17	1200
391	1180		LMT-01	LeMon Pizza Tablet	-1		17	1201
391	1181		LMT-02	Pizza Tablet Set	-1		17	1204
391	1182		Z00004B-5	Tablet PC Green Set - B5	-1		17	1205
391	1183		Z00002	Tablet PC 64GB White	13	1051	17	1209
391	1183	1	Z00002	Tablet PC 64GB White	13	1051	17	1209
392	1184		S10000	Server Point 10000	-1		17	1209

On this table, there is no golden arrow on the links. Since we can't tell where something leads it does not have a link. In order to get a golden arrow, you must join tables together, and use its docEntry.

```
SELECT t1.docentry, T0.[DocEntry], T0.[LineNum], T0.[ItemCode],
T0.[Dscription], T0.[TargetType], T0.[TrgetEntry],
T0.[BaseType], T0.[BaseEntry] FROM DLN1 T0
INNER JOIN ORDR T1 ON T0.baseEntry = t1.docentry
```

This gives us a table with a golden arrow to the sales order these delivery items came from.

#	Internal Number	Document Internal ID	Row Number	Item No.	Item/Service Description	Target Document Type	Target Document Internal ID	Base Document Type	Base Document Internal ID
384	1194	1174		LMT-02B	Pizza Tablet Set - Batch	-1		17	1194
384	1195	1175		Z-MRP1	Penguin Tablet	-1		17	1195
384	1196	1176		Z-MRP1	Penguin Tablet	-1		17	1196
384	1197	1177		Z-MRP1	Penguin Tablet	-1		17	1197
384	1198	1178		Z-MRP1	Penguin Tablet	-1		17	1198
384	1200	1179		Z-02	Penguin Tablet Set	-1		17	1200
384	1201	1180		LMT-01	LeMon Pizza Tablet	-1		17	1201
385	1204	1181		LMT-02	Pizza Tablet Set	-1		17	1204
385	1205	1182		Z00004B-5	Tablet PC Green Set - B5	-1		17	1205
385	1209	1183		Z00002	Tablet PC 64GB White	13	1051	17	1209
385	1209	1183	1	Z00002	Tablet PC 64GB White	13	1051	17	1209
385	1209	1184		S10000	Server Point 10000	-1		17	1209

I can expand this to make a report with customer and docnumber

```
SELECT t1.docnum AS "S/O #", t1.cardcode AS "Customer"
T0.[DocEntry], T0.[ItemCode], T0.[Dscription], T0.[TargetType],
T0.[TrgetEntry], T0.[BaseType], T0.[BaseEntry] FROM DLN1 T0
INNER JOIN ORDR T1 ON T0.baseEntry = t1.docentry
WHERE t0.lineNum = 0 --this is a hack and may be inaccurate if
items split from here
```

Notice the highlighted rows listing a target type and an ID. Those are numerical identifiers for modules. There are hundreds of these but here's a short list of them

Table#	Table	Name	Link
13	OINV	A/R Invoice	DocEntry
14	ORIN	A/R Credit Memo	DocEntry
15	ODLN	Delivery	DocEntry
16	ORDN	Returns	DocEntry
17	ORDR	Sales Order	DocEntry
18	OPCH	A/P Invoice	DocEntry
19	ORPC	A/P Credit Memo	DocEntry
20	OPDN	Goods Receipt PO	DocEntry
21	ORPD	Goods Return	DocEntry
22	OPOR	Purchase Order	DocEntry
23	OQUT	Sales Quotation	DocEntry
24	ORCT	Incoming Payment	DocEntry

Multiple Joins

You often need more than one table. You can use more than one join to do this. Let's look at one example.

We'll start with some basic information from the sales order.

```
SELECT T0.[DocNum], T0.[CardCode], T0.[DocTotal]  FROM ORDR
```

Run and you get every sales order. Let's only get today's sales orders. Edit the query to add

```
T0 WHERE t0.DocDate = GETDATE()
```

Run and you will have a Surprise. There are no sales orders today.

There is a type bug we didn't take into consideration. We think of dates and just dates. However, in SQL they are dates and times. So the WHERE clause is looking for a date at exactly that second in time.

We need to strip the time off GETDATE. We can do that with the CONVERT:

```
T0 WHERE t0.DocDate = CONVERT(date,getdate())
```

Run and we get todays orders.

While we have the CardName in ORDR, we don't have the customer's balance. I'll join ORDR to get those

```
SELECT T0.[DocNum], T0.[CardCode], T1.CardName, T1.Balance,
T0.[DocTotal]
FROM ORDR T0
INNER JOIN OCRD t1 ON T0.CardCode = t1.CardCode
WHERE t0.DocDate = CONVERT(date,GETDATE())
```

Run and we get information about the customer.

You can add more joins to the end of an existing join. For example, Lets add the price list name, which is stored in OPLN.listname. I have the price list for the customer in OCRD.listnum.

```
SELECT T0.[DocNum], T0.[CardCode], T1.CardName, T1.Balance,
t2.listname, T0.[DocTotal]
FROM ORDR T0
INNER JOIN OCRD t1 ON T0.CardCode = t1.CardCode
INNER JOIN OPLN t2 ON t1.listNum = t2.listNum
WHERE t0.DocDate = CONVERT(date,GETDATE())
```

So far, we've done one to one joins. Let's do a one to many. Let's break out the line items for the sales order using docentry.

```
SELECT T0.[DocNum], t3.linenum, T0.[CardCode], T1.CardName,
T1.Balance, t2.listname, t3.ItemCode,t3.quantity,
t3.price,t3.linetotal,  T0.[DocTotal]
FROM ORDR T0
INNER JOIN OCRD t1 ON T0.CardCode = t1.CardCode
INNER JOIN OPLN t2 ON t1.listnum = t2.listnum
Inner JOIN RDR1 t3 ON t0.docentry = t3.docentry
WHERE t0.DocDate = CONVERT(date,GETDATE())
```

Again that works okay. However, the data repeats if there is more than one item in the order.

One last one for this lesson. I'd like some inventory information from OINV.

```
SELECT T0.[DocNum], T0.[CardCode], T1.CardName, T1.Balance,
t2.listname,t3.linenum, t3.ItemCode,t4.ItemName,t4.iscommited
,t4.onHand, t3.quantity, t3.price,t3.linetotal,  T0.[DocTotal]
FROM ORDR T0
INNER JOIN OCRD t1 ON T0.CardCode = t1.CardCode
INNER JOIN OPLN t2 ON t1.listnum = t2.listnum
INNER JOIN RDR1 t3 ON t0.docentry = t3.docentry
INNER JOIN OITM t4 ON t3.itemCode = t4.Itemcode
WHERE t0.DocDate = CONVERT(date,GETDATE())
```

And that gives us our report.

Now a word of warning: once you do a one to many Join, you have to be careful. For now, only to a one to one join if you want to join to the table. Inventory items are one to one to item codes in RDR1. So that's okay.

For now, stay away from base and target documents. They can be one to many relationships.

We'll discuss multiple one to many joins next time.

Assignment: add two or more joins to a report.