

# The V4 Challenge for SQL and OLAP

## Introduction

This paper presents five challenges for your data management tools. Can you quickly and easily solve these problems using one or more of the tools in your data management toolkit? Solutions to each of the problems is given using V4<sup>®</sup>, a revolutionary new database tool, developed and supported by MKS Inc.

All of the problems presented are based on relational tables taken from actual applications. Only the necessary columns required for each particular challenge are listed. It is assumed that the data is in relational form. If an OLAP product is to be used, then your “solution” should include the effort required to map from flat tables to the multi-dimensional cube.

V4 uses new concepts. Its syntax may appear foreign to you. The point here is *not* to explain the basics of V4 or the particulars of a solution. What we *want* to show is how easily and quickly one can create solutions using V4. Note that each challenge presented has been implemented in V4 for a real client. The results shown here are based on real data, but some numbers and descriptions have been altered due to the proprietary nature of the data. The time to implement ranged from about 30 minutes for Challenge II to 8 hours for Challenge V. You may not want to actually implement a solution to all of these challenges, but at a minimum, estimate the level of effort that would be required.

These challenges are about manipulating data, not about providing a pretty GUI front-end or generating multi-color reports. Most products, including V4, directly or indirectly provide these ease-of-use facilities. The example reports shown below are images taken directly from Microsoft Excel using the V4 Add-In.

## Challenge I – Ranking Of Sales Reps By Month and Sales Manager

This challenge takes information from two tables. The first, Sale, is a record of each sale. The columns in the table are Date (date of the sale), SaleAmount (dollar amount of the sale), and Emp (the sales representative responsible for the sale). The second table, Emp, contains a record for each employee. Employees include sales representatives and their managers. The columns are Id (identifier of the employee), Name (the employee’s name), and Mgr (the employee who is the manager of the sales rep).

```
Sale(Date SaleAmount Emp)
Emp(Id Name Mgr)
```

The required output shows months across and managers down. For each manager and month, the top five sales representatives are shown as well as their corresponding sales for that month (i.e. two columns for each month). The top sales rep for each month is highlighted in red. Note that any range and/or list of months may be specified, and that a manager may not have five sales reps reporting to him/her.

```
Bind [Rank Month.. Emp.. Int..] List(Sort([Emps] Reverse::[Sales],0) Nth::Int*)
Bind [TopRep Month..] List(Sort(Emp.. Reverse::[Sales],0) Nth::1)
Bind [RankedRep Month.. Emp..] If([Rank Int:1] = [TopRep]) TopEmp:=[Rank Int:1] Id [Rank Int:1] Id]
Bind [RepRank Month..]
Do( Tally(Emp.. (ListOf::Emp* By::Emp.Mgr ByList::[MgrList] Bind::[Emps]))
  Tally(Sale.. If::[Sale.Date == Month*) (Sum::Sale.SaleAmount By::[Sale.Emp DTInfo(Sale.Date Month?)]) Bind::[Sales])
  SSDim(Dim:TopEmp Style::"Color:Red,Italic")
  EchoS(" "" TC:=Str("{2}" Month*)/Month*) EchoS(" TR:"Rank" (TR:"Rep" TR:"$ Sales")/Month*)
  Sort([MgrList] By::Emp.Id
  Do::(EchoS(Emp.Name 1 ([RankedRep] [[Rank Int:1] Sales],0)/Month*)
    Enum(Int:2..5 If::DefQ([Rank Int*]) @EchoS(" Int* ([[Rank Int*] Id] [[Rank Int*] Sales],0)/Month*) EchoS(0) )
  EchoS(End::"SalesRep Ranking by Manager") )
```

Evaluating [RepRank Month:Jul-99..Dec-99] would result in output looking like-

SalesRep Ranking by Manager

	Rank	Jul-99		Aug-99		Sep-99		Oct-99		Nov-99		Dec-99	
		Rep	\$ Sales	Rep	\$ Sales	Rep	\$ Sales	Rep	\$ Sales	Rep	\$ Sales	Rep	\$ Sales
John Jones	1	1270	102,650	1115	981,629	1490	2,980,837	1115	9,625,157	1675	11,158,476	1490	2,064,797
	2	1140	21,670	1605	731,736	1115	2,891,326	1490	5,508,607	1490	8,064,163	1115	893,124
	3	3100	0	1490	439,619	1675	891,742	1270	4,613,454	1605	6,968,765	1675	549,910
	4	3145	0	1675	292,236	1850	681,189	1140	2,422,023	1140	4,140,211	1140	475,000
	5	1625	0	1850	126,522	1390	246,230	1605	1,740,138	1115	3,208,949	1270	303,438
Wally Smith	1	1500	134,343	1855	4,216,489	1165	41,940,057	1500	57,317,377	1905	66,803,065	1655	6,775,662
	2	1905	51,039	1565	4,100,525	1370	40,289,652	1565	50,209,253	1405	38,173,667	2060	6,541,177
	3	1845	47,466	1405	2,471,054	1405	33,376,297	1405	40,729,151	1855	30,036,546	1065	5,345,990
	4	2280	12,995	1500	2,440,672	1565	20,517,811	1165	33,419,382	1805	22,853,455	1165	4,062,919
	5	1165	11,720	1165	2,205,099	1805	19,264,733	1575	20,569,501	1165	15,980,363	1805	3,862,947
Harry Palm s	1	1915	2,899,377	1205	13,128,604	1778	42,262,014	1915	35,258,875	1790	54,413,519	1890	9,728,729
	2	1600	1,290,989	1455	5,593,805	1890	16,880,230	1325	34,342,933	1890	43,572,783	1620	5,870,027
	3	1455	1,206,287	1790	4,714,275	1670	15,638,264	1060	30,051,179	1775	28,224,432	1775	5,698,236
	4	1325	1,070,676	1325	2,816,268	1205	15,393,483	1890	26,718,209	1325	27,198,961	1060	5,275,046
	5	1865	613,600	1590	2,392,232	1455	10,398,032	1205	26,639,700	1865	22,982,546	1865	3,827,444
Fran Little	1	2295	1,433,561	1095	7,347,290	1360	69,315,485	1095	71,055,034	2225	85,149,140	1185	14,250,617
	2	1395	424,722	1360	5,202,334	1215	45,158,871	1360	45,594,126	2230	33,601,786	2295	10,451,196
	3	1530	380,160	1085	3,446,402	1770	25,360,193	1215	38,832,833	1395	21,847,625	1505	9,412,532
	4	2230	359,673	1185	3,337,344	1745	15,775,273	3085	31,827,724	1095	20,862,312	2230	9,175,107
	5	1145	234,598	1215	3,124,003	1085	13,026,763	1185	26,516,644	3075	19,795,727	1085	8,208,381

# The V4 Challenge for SQL and OLAP

## Challenge II – Extract Lists of Customers Based on Keywords in Name

Given a list of keywords, scan through the customer table, first extracting all customers whose name contains the first keyword, then customers whose name contains the second (but not the first), and so forth for all keywords. The number of keywords is variable with a maximum of 100 keywords. The single table required for this analysis is–

Cus(Id Name City State)

The single V4 rule required to perform this is shown below–

```
Bind [CusGroups List.]
Do(Project(Dim:Cus Dim:Alpha Area::1 Unique::0) Project(Dim:OK Dim:Cus Point::OK:True)
Enum(Cus.. @Enum(Str(Cus.Name ListOf:"" Trim?) @Project(Cus* Alpha*))
Enum(List* @Enum(Project(Dim:Cus Str(Nld*)),0) If::OK:=Cus* @Do(Project(OK:False Cus*) EchoT(Nld* Cus.Id .Name .City .State))))
```

To run this, extracting all customers with “Band”, then “Orchestra”, then “Music” would be done by simply evaluating “[CusGroups (Band Orchestra Music)]”. The output would be a tab delimited file with columns for the keyword, the customer Id, the customer name, city and state.

## Challenge III – Variable Bucket Accounts Receivable Aging

The challenge is to generate what is called an Aging Report. It looks at all invoices with balances due, aggregates by the sales representative responsible for the sale, and “ages” the totals. Aging is done by determining how old the invoice is, or how long the balance has been due. For example, an invoice dated 1-Feb-00 would be considered “over 30 days old” as of 10-Mar00. Two tables are required. The Emp table gives the name and identifier of each sales rep. The Inv(oice) table contains the current balance due, the employee (sales rep), and invoice date. The analysis should handle variable number of aging buckets.

Emp(Id Name)  
Inv(BalanceAmt Emp InvDate)

The V4 rules to perform this analysis are shown below. Evaluating “[CusAgeByRep Periods:30,45,60,90,120 Date:1-Jun-00]” generated the report immediately following the V4 rules.

```
Bind [AgeTallyArgs Periods.. Date..]
MakeL(Sum::Inv.BalanceAmt By::Inv.Emp MakeP(Tag:Bind MakeL(AgeAmt Periods*))
MakeP(Tag:IfOnce MakeL(IntMod:LE @[Inv.InvDate] {Date* - Periods*}))/Sort(Periods* Reverse::Periods*)
Bind [Func:CusAgeByRep Periods.. Date..]
Do(Tally(Inv.. (Sum::Inv.BalanceAmt By::Inv.Emp Bind::[ARBalance] ByList::[RepList])
[AgeTallyArgs] (Sum::Inv.BalanceAmt By::Inv.Emp Bind::[AgeCur])
EchoS(T:"SlsRep" :Name TR:"Balance" :Current TR:=Str("Over " Periods*)/Periods* Echo::0)
Enum(Sort([RepList] By::Emp.Id) @EchoS(Emp.Id .Name .ARBalance .AgeCur,0 [AgeAmt],0/Periods*))
End::(1 EchoS(" "Grand Total" RCV(3) RCV() RCV()/Periods*))
EchoS(End::Str("Customer Aging By SlsRep As of " Date")) )
```

Customer Aging By SlsRep As of 01-Jun-00

SlsRep	Name	Balance	Current	Over 30	Over 45	Over 60	Over 90	Over 120
1005	John Smith	17,428.73	-99.30	0	17,130.05	384.30	0	13.68
1010	Harry Jones	5,904.91	810.00	0	2,962.18	958.44	228.56	945.73
1015	William Johnson	39,993.85	4,749.75	7,955.33	21,702.51	5,392.26	0	194.00
1020	Melinda Wolosey	3,795.15	108.30	0	1,815.10	0	0	1,871.75
1035	Peikup Andropov	4,960.51	2,440.51	2,520.00	0	0	0	0
1040	Fred Harrison	5,000.51	1,586.50	1,013.22	2,012.21	93.03	0	295.55
1045	Sally Bodine	27,668.84	2,110.71	131.29	7,120.86	11,622.90	6,228.53	454.55
1050	Mary Contrary	193.27	0	0	0	0	0	193.27
1060	Ralph Simms	60,860.07	15,685.01	7,730.48	30,491.13	6,953.46	0	0
	Grand Total	165,805.84	27,391.48	19,350.32	83,234.04	25,404.38	6,467.09	3,968.53

## Challenge IV – Number of Unique Customers Purchasing Items

This challenge scans purchases or sales for two separate time periods. The number of *unique* customers purchasing each item in the two time periods is calculated and output. The output is sorted by the vendor supplying the inventory and then by item. It is often the case that the most elegant solution is not the most efficient. This challenge is computationally expensive due to the requirement for unique customers. The example solution given below was run off of 3.9 million sales records in less than 7.5 CPU minutes on a 400 megahertz PC. Over eight thousand lines of output were generated in the actual report. Only a few are shown here as an example.

The three tables required are shown below. The Ven(dor) table contains the name and identifier for each vendor. The IM (inventory) table has the identifier, name, and vendor of each item. The Sale(s) table contains the customer, inventory item, and time period of each sale.

## The V4 Challenge for SQL and OLAP

Ven(Id Name)  
 IM(Id Name Ven)  
 Sale(Cus IM Period)

Evaluating “[NumCusVen UP1:9901..9903 UP2:9907..9909]” results in the report shown following the V4 rules.

```
Bind [NumCusVen UP1.. UP2..]
Do(Tally(Sale.. (By::Sale.IM ByList::[NIMList])
  (UCount::Sale.Cus By::Sale.IM Bind::[CusCount1] If::{Sale.Period == UP1*})
  (UCount::Sale.Cus By::Sale.IM Bind::[CusCount2] If::{Sale.Period == UP2*}))
EchoS(T:"Id" : "Description" TR:"Period 1" : "Period 2" Echo::0)
Enum(Sort([NIMList] By::IM.Ven By::IM*) @EchoS(IM* IM.Desc [CusCount1],0 [CusCount2],0)
  After::((IM.Ven EchoS(" " Str("Vendor " IM.Ven.Id " - " IM.Ven.Name))))
EchoS(End::Str("Number NABCA Licensees Purchasing Product\Period " UP1* " versus " UP2*)) )
```

*Number NABCA Licensees Purchasing Product  
 Period 1999/01..1999/03 versus 1999/07..1999/09*

<i>Id Description</i>	<i>Period 1</i>	<i>Period 2</i>
<i>Vendor 1585 - CLICQUOT INC</i>		
5078 CLICQUOT YELLOW LAB	0	13
6924 CLICQUOT YELLOW LAB	119	3
<i>Vendor 1610 - CLOS DU VAL WN</i>		
4277 CLOS DU VAL CARN CHA	18	21
6196 CLOS DU VAL CAB SAUV	74	70
<i>Vendor 1618 - CLOVER HLL VNYD</i>		
6957 CLOVER HILL SPICED A	25	34
7976 CLOVER HILL CONCORD	1	4
7977 CLOVER HILL PINK CAT	0	2
<i>Vendor 1641 - CODERA WINE GRP</i>		
8375 BLACKSTONE MERLOT	43	50
<i>Vendor 1643 - CODORNIU USA</i>		
4713 CODORNIU BRUT	20	19
4826 CODORNIU BRUT	60	40
6642 CODORNIU NAPA BRUT	3	2
7920 CODORNIU BLC DE BLCS	0	1

## Challenge V – Raw Materials Requirements Based on Sales Projection

This is the last and most complex challenge. The problem is to take weekly sales projections of finished goods, explode them through bills-of-material and generate a consolidated list of raw materials needed to satisfy the projection. Additionally, the onhand quantities of the raw materials and the projected receipts (total quantity and detail) from open purchase orders should be shown. Raw material shortages should also be shown by week.

Seven tables necessary for this analysis. The IM table contains entries for each inventory item (identifier and description). The IML table contains entries giving the on-hand quantity (uOHQty) for each item at each warehouse location (Loc). The PO and POD tables describe open purchase orders with detail for each outstanding inventory item- the quantity remaining to be received (uRemainQty) and the Expected due date. The bill-of-material (BOM) table has an entry for each finished good and the BOMD has entries for each raw material component needed to make the finished product. The Yield is the amount/quantity of the finished product. The BOMIM is the component raw material, and BOMQty is the quantity/amount of the raw material. The SPW table has the weekly projected sales for the finished goods. Note that in the “real world” these quantities may be given in different units. The recipe in the bill-of-material may specify a quantity in grams, but the warehouse quantity (in the IML table) may be in kilograms or pounds.

IM(Id Desc)  
 IML(IM Loc uOHQty)  
 PO(Id Loc)  
 POD(PO IM uRemainQty Expected)  
 BOM(Id IM Yield)  
 BOMD(BOM ParentIM BOMIM BOMQty)  
 SPW(Id UWeek uQty IM)

The rules to perform this analysis are shown below. To run the project for four weeks beginning with 29-May-00, for warehouse location Main on finished goods projection Test one would evaluate “[ProjRMSPW Week:29-May-00..19-Jun-00 Loc:Main SPWId:Test]”.

# The V4 Challenge for SQL and OLAP

```

Bind [Explode IM.. Week.. uIMQty..]
DefQ(IM.BOMList
  @Enum(IM.BOMList @[Explode BOMB.BOMIM uIMQty:=(BOMB.BOMQty * (uIMQty* / BOMB.BOM.Yield))])
  @Do(Context(MakeP(Dim:Exp)) BindQE(Exp.IM IM*) BindQE(Exp.uIMQty uIMQty*) BindQE(Exp.Week Week*))
Bind [CRMQty IM.. Week..] Sum([RMQty],0/List(List(Week**) To::Week*))
Bind [ProjRMSPW Week.. Loc.. SPWId..]
Do(Tally(IML.. If:([IML.Loc == Loc*]) (Sum::IML.uOHQty By::IML.IM Bind::[uOH] )
  Tally(POD.. Parent::POD.PO Plf:({PO.Loc == Loc*}) If:({POD.Expected == Week*})
    (ListOf::POD* By::POD.IM Bind::[RcvList]) (Sum::POD.uRemainQty By::POD.IM Bind::[uOO]))
  Tally(BOMB.. Id::BOMB (ListOf::BOMB* By::BOMB.ParentIM Bind::[BOMList]))
  Tally(SPW.. If:({SPW.Id == SPWId* & SPW.Week == Week*})
    (Sum::SPW.uQty By::SPW.IM Bind::[TPrjQty])
    (Sum::SPW.uQty By::([SPW.IM SPW.UDate.Week Bind::[PrjQty]) (By::SPW.IM ByList::[IMList]))
  Enum([IMList] If::Def(@IM.BOMList) @Enum(Week* If::DefQ(IM.PrjQty) @[Explode IM* Week* uIMQty:=IM.PrjQty])
  Tally(Exp.. (Sum::Exp.uIMQty By::([Exp.IM Exp.Week Bind::[RMQty]) (By::Exp.IM ByList::[RMLList]))
  EchoS(" " " " " " TC:=Str("(" ListSize(Week*) " )Needed" TC:=Str("(" ListSize(Week*) " )Short")
  EchoS(TDD:"Id" T:"Description" TR:"OnOrder" TR:"OnHand" (Week.UDate)/Week* (Week.UDate)/Week* Echo::0)
  Sort([RMLList] By::IM.Id
  Do:([EchoS(IM.Id IM.Desc IM.uOO," IM.uOH," [RMQty]," /Week*
    @GE(IM.uOH,0 [CRMQty Week*] " " Minimum([RMQty],0 {[CRMQty Week*] - IM.uOH,0))/Week* )
    Enum([RcvList],0) @EchoS(" " Str(" PO# " POD.PO.Id " on " POD.Expected " for " POD.uRemainQty))) )
  EchoS(End::Str("Raw Materials Requirements Projection\ISchedule " SPWId*))
)

```

Raw Materials Requirements Projection  
Schedule Test

Id	Description	OnOrder	OnHand	Needed				Short			
				5/29/00	6/5/00	6/12/00	6/19/00	5/29/00	6/5/00	6/12/00	6/19/00
10060B	PE ANUTS		0ea			6000ea				6,000	0
10316C	COOKIES, CHOC SANDWICH BU		0ea			22040ea	16040ea			22,040	16,040
10525I	PRETZELS, BULK	320lb	196lb		0.656lb	66lb	22lb				
	RASPBERRY JAM										
10841I	DEXTROSE		0lb			4643.2lb	4643.2lb			4,643	4,643
10845D	CHOCOLATE, MILK		0ea		24019.2ea	12009.6ea	12009.6ea		24,019	12,010	12,010
10924B	CHOCOLATE, DARK		1275ea		24240ea	14140ea			22,965	14,140	0
10933B	SUGAR, 6X		0ea			16160ea	16160ea			16,160	16,160
10935B	WATER		0ea			9800ea				9,800	0
11321C	TRAY 8 CAVITY		0ea			9800ea				9,800	0
11324I	TRAY 7 CAVITY COOKIE		0lb			371.2lb	371.2lb			371	371
11415A	TRAY 9 CAVITY ROUND		0ea			443.8ea	332.8ea			444	333
11659I	TRAY 7 CAVITY SQUARE COOK		0lb		0.7lb				1	0	0
12025E	CECO HADD 4 OZ MINTS		0ea			2211ea				2,211	0
12220A	CECO HADD 4 OZ COCONUT		350ea	3206.4ea	1977.8ea	1370.88ea	2178ea	2,856	1,978	1,371	2,178
12244C	CECO HDF 4OZ PB PATTIES		0ea		21052.5ea		21052.5ea			21,053	0
12245D	CARTON BULK ROSE 72 CT.		0ea		10508.4ea		10508.4ea			10,508	0
12540I	CARTON HADD 4 OZ BOX		0lb		99lb	112.75lb	153lb			99	113
12558C	CARTON (404) OPEN TOP		0ea		18045ea	18045ea	18045ea			18,045	18,045
12559C	EASYFOND	186000ea	0ea	192480ea	72180ea	36060ea	108270ea	192,480	72,180	36,060	108,270
	PO# 12691 on 10-Jul-00 for 36000ea										
	PO# 12691 on 30-May-00 for 30000ea										
	PO# 12691 on 01-Jun-00 for 72000ea										
	PO# 12691 on 06-Jun-00 for 48000ea										
12582A	CARTON, FLUTED OVAL TIN		21ea		874.65ea		874.65ea		854	0	875
12633B	CECO, SIGNATURE TRUFFLES		0ea		18180ea	18180ea	18180ea		18,180	18,180	18,180