

Trying to find the best coverage match from table 2 for tables 1s VIN. The red boxed statement is the answer.

Table 1 (tbl1)

VIN (V)	MONTHS (MN)	MILEAGE (MI)
1N4AL3AP0AN342561	42	77,433

Table 2 (tbl2)

VIN (V)	COVERAGE (CVG)	MONTHS (MN)	MILEAGE (MI)
1N4AL3AP0AN342561	A	12	12,000
1N4AL3AP0AN342561	B	36	36,000
1N4AL3AP0AN342561	C	60	50,000
1N4AL3AP0AN342561	D	72	999,999

Logic Steps:

If tbl1.MI > tbl2.CVG.D MI and tbl1.MN < tbl2. CVG.D MN

Then tbl2.CVG = D

Else

If tbl1.MI < tbl2. CVG.D MI and tbl1.MN > tbl2. CVG.D MN

Then tbl2.CVG = D

Else

If tbl1.MI > tbl2. CVG.D MI and tbl1.MN > tbl2. CVG.D MN

Then tbl2.CVG = D

Else

If (tbl1.MI > tbl2. CVG.C MI and tbl1.MN < tbl2. CVG.C MN)
and (tbl1.MI < tbl2.CVG.D MI and tbl1.MN < tbl2. CVG.D MN)

Then tbl2.CVG = C

Else

If (tbl1.MI < tbl2. CVG.C MI and tbl1.MN > tbl2. CVG.C MN)
and (tbl1.MI < tbl2.CVG.D MI and tbl1.MN < tbl2. CVG.D MN)

Then tbl2.CVG = C

Else

If (tbl1.MI > tbl2. CVG.C MI and tbl1.MN > tbl2. CVG.C MN)
and (tbl1.MI < tbl2.CVG.D MI and tbl1.MN < tbl2. CVG.D MN)

Then tbl2.CVG = C

Else

If (tbl1.MI > tbl2. CVG.B MI and tbl1.MN < tbl2. CVG.B MN)
and (tbl1.MI < tbl2.CVG.C MI and tbl1.MN < tbl2. CVG.C MN)

Then tbl2.CVG = B

Else

If (tbl1.MI < tbl2. CVG.B MI and tbl1.MN > tbl2. CVG.B MN)
and (tbl1.MI < tbl2.CVG.C MI and tbl1.MN < tbl2. CVG.C MN)

Then tbl2.CVG = B

Else

If (tbl1.MI > tbl2. CVG.B MI and tbl1.MN > tbl2. CVG.B MN)
and (tbl1.MI < tbl2.CVG.C MI and tbl1.MN < tbl2. CVG.C MN)

Then tbl2.CVG = B

Else

If (tbl1.MI > tbl2. CVG.A MI and tbl1.MN < tbl2. CVG.A MN)
and (tbl1.MI < tbl2.CVG.B MI and tbl1.MN < tbl2. CVG.B MN)

Then tbl2.CVG = A

Else

If (tbl1.MI < tbl2. CVG.A MI and tbl1.MN > tbl2. CVG.A MN)
and (tbl1.MI < tbl2.CVG.B MI and tbl1.MN < tbl2. CVG.B MN)

Then tbl2.CVG = A

Else

If (tbl1.MI > tbl2. CVG.A MI and tbl1.MN > tbl2. CVG.A MN)
 and (tbl1.MI < tbl2.CVG.B MI and tbl1.MN < tbl2. CVG.B MN)
 Then tbl2.CVG = A

```
select *,
switch (
[CVG-ITM] = 'FDECDFL', 1,
[CVG-ITM] = 'PWRTRN', 2,
[CVG-ITM] = 'BASIC', 3,
[CVG-ITM] = 'CARB', 4
) as [Priority]
from ee_records r
left join ee_record_possiblities p on p.vin = r.vin
```

```
Switch (Nz ([FiveYearAVE],0)<100000,1,[FiveYearAVE]<250000,2,[FiveYearAVE]<500000,
3,[FiveYearAVE]<1000000,4,[FiveYearAVE]<2000000,5,[FiveYearAVE]<5000000,6,[FiveY
earAVE]<10000000,7,[FiveYearAVE]<25000000,8,[FiveYearAVE]<50000000,9,[SumOfYR5]=
50000000,10,[ThreeYearAVE]>50000000,10)
```

See attached sample file. The first query uses the Switch function which evaluates a sum.

The second query uses the Switch function but is based on the first query.

Both queries request parameter values. Any recommendation on how to proceed?

[SwitchTest.accdb](#)

With aggregate queries, unless you have saved the output into tables where your sumofwhatevers are actual field names as opposed to aliases for sums, averages etc -- you need to use the expression itself.

Long winded explanation, but it looks like this:

```
SELECT tblMarketSegment.MarketSegment,
Sum(tblOppProj.EstimatedOpportunityRevenue1) AS
SumOfEstimatedOpportunityRevenue1,
Switch(Sum(tblOppProj.EstimatedOpportunityRevenue1) <100000,1,100000<=
Sum(tblOppProj.EstimatedOpportunityRevenue1)
<250000,2,250000<=Sum(tblOppProj.EstimatedOpportunityRevenue1)
<500000,3,500000<=Sum(tblOppProj.EstimatedOpportunityRevenue1)
<1000000,4,1000000<=Sum(tblOppProj.EstimatedOpportunityRevenue1)
<2000000,5,2000000<=Sum(tblOppProj.EstimatedOpportunityRevenue1)
<5000000,6,5000000<=Sum(tblOppProj.EstimatedOpportunityRevenue1)
<10000000,7,10000000<=Sum(tblOppProj.EstimatedOpportunityRevenue1)
<25000000,8,25000000<=Sum(tblOppProj.EstimatedOpportunityRevenue1)
<50000000,9,50000000<=Sum(tblOppProj.EstimatedOpportunityRevenue1) ,10) AS
RevenueIndex
```

```
FROM tblOppProj INNER JOIN tblMarketSegment ON tblOppProj.lkOppProjMarket =  
tblMarketSegment.MarketSegmentID  
GROUP BY tblMarketSegment.MarketSegment;
```

Hmm... I blindly reused your SQL in the above correction.

The following general syntax will not work:

Number1 <= Field1 < Number2

The correct syntax is either (see my post at <http:#a39631380>):

Field1 BETWEEN Number1 and (Number2 -1)

Or this:

Field1 >= Number1 AND Field1 < Number2

Or to use the implied sequential order of evaluation that fyed mentioned in his post.

I personally prefer to explicitly include all endpoints... it just makes the flow/logic look clearer to me.

... and you also need to use the expression SUM(x) rather than the alias SumOfx

Correcting the SQL using BETWEEN:

```
SELECT  tblMarketSegment.MarketSegment,  
Sum(tblOppProj.EstimatedOpportunityRevenue1) AS  
SumOfEstimatedOpportunityRevenue1,  
Switch(Sum(tblOppProj.EstimatedOpportunityRevenue1) <100000,1,  
Sum(tblOppProj.EstimatedOpportunityRevenue1) BETWEEN 100000 AND 249999 ,2,  
Sum(tblOppProj.EstimatedOpportunityRevenue1) BETWEEN 250000 AND 499999,3,  
Sum(tblOppProj.EstimatedOpportunityRevenue1) BETWEEN 500000 AND 999999,4,  
Sum(tblOppProj.EstimatedOpportunityRevenue1) BETWEEN 1000000 AND 1999999,5,  
Sum(tblOppProj.EstimatedOpportunityRevenue1) BETWEEN 2000000 AND 4999999,6,  
Sum(tblOppProj.EstimatedOpportunityRevenue1) BETWEEN 5000000 AND 9999999,7,  
Sum(tblOppProj.EstimatedOpportunityRevenue1) BETWEEN 10000000 AND 24999999,8,  
Sum(tblOppProj.EstimatedOpportunityRevenue1) BETWEEN 25000000 AND 49999999,9,  
Sum(tblOppProj.EstimatedOpportunityRevenue1) >= 50000000, 10)  
AS RevenueIndex  
FROM tblOppProj INNER JOIN tblMarketSegment ON tblOppProj.lkOppProjMarket =  
tblMarketSegment.MarketSegmentID  
GROUP BY tblMarketSegment.MarketSegment;
```