As with the first exam, my first step was to break down the given task into chunks. After reading Part 1 and looking at the data; link the names of each customer to each invoice; determine if each customer was a repeat for each part; and finally return each part ordered in each invoice excluding the original part number. I determined that there were four distinct tasks I needed to accomplish: how to extract the invoices for each part, The first chunk was to determine a way to extract invoice numbers from sales data.

First I created a data validation list that lists all part number in Parts Data. I sorted this by category, then manufacturer, and then finally part number. I felt that this would be the most useful for the user. I determined that invoice was the best place to start the table because of how the data was stored in Sales Data. In Sales Data, the invoices are itemized. This means that each individual order is split up by each part that was ordered in a single sale. For example, invoice 0000019 is divided among six entries for each of the six different part types ordered in that sale. Using this observation, I used an index function that would return the invoice number that corresponded with each entry the part was ordered. The final function was:

=IFERROR(INDEX(InvoiceNo,SMALL(IF(ItemCode=$D$5,ROW(InvoiceNo)-1),ROW(1:1))),"")

The next two chunks were relatively strait forward. To retrieve customer name, I used an index function to match the invoice number to customer name. The final function was:

=IF(D7="","",INDEX(BillToName,MATCH(D7,InvoiceBill,0)))

For the third chunk, I created an additional column Sales Data that consisted of an index function to retrieve customer name from Bill to Data using invoice number. Then I used a sum array function that counted the entries that contained the desired part number and customer name. the final function was:

=IF(B7="","",SUM((B7=CustName)\*(ItemCode='Model 1'!$D$5)\*1))

The final chunk of this part required me to extract each part ordered alongside the desired part of invoice. I decided to use a multi-criteria index array to return each unique part ordered. The final function was:

=IFERROR(IF(SUM((InvoiceNo=$D7)\*1)<COUNTA($E$6:E$6),"",INDEX(ItemCode,SMALL(IF($D7=InvoiceNo,IF(ItemCode<>$D$5,ROW(ItemCode)-1)),COUNTA($E$6:E$6)))),"")

I decided to use a COUNTA function instead of a ROW function because it would work going left to right.

After each of the chunks was completed for part one, I decided to do some cleaning up. I added the IFERROR function to the first part so that when the function could not retrieve any more values, a blank would appear. I also added the initial IF functions to the other two chunks to cut down on total calculation time be returning a blank if invoice number was blank. To also cut down on calculations, I determined the amount of rows and columns the table needed by counting the number of entries for each part and the number of entries for each invoice. I determined 98 rows and 38 item columns were necessary. I decided to round them up to 100 and 40 respectively because I prefer round numbers. I finally added the data validation for each part number and the index functions for manufacturer and category for the desired part. With this complete I moved on to Part 2.

Part 2 was a much more difficult part. After reading the guidelines and looking at the example, I determine that there were three chunks. These were, retrieving a unique list of parts from Model 1, determining how frequently each part was bought with the desired part, and retrieving the manufacturer and category for each part. The first chunk was by far the most difficult. I had to form a unique list of parts from multiple columns in Model 1. After some trial and error, I determined that index functions and usual frequency and sum product function would not work across multiple columns, I had to use a different formula. I eventually arrived at the formula:

=IF(ROW(1:1)>$B$5,"",INDIRECT("'Model1'!"&TEXT(MIN(IF((Items<>"")\*(COUNTIF($C$6:C6,Items)=0),ROW('Model 1'!$E$7:$E$105)\*100+COLUMN('Model 1'!$E$6:$AR$6))),"R000C00"),FALSE))

This function allows me to return a unique list from multiple columns.

The second chunk was relatively straight forward. To determine how many times the part was ordered, I used a sum array to count then number of times each part appeared in Model 1. The formula I used was:

=IF($C7="","",SUM((Items=$C7)\*1))

To find the number of repeat and single customers I used the formulas:

=IF($C7="","",SUM((Items=$C7)\*(Unique>1)\*1))

and

=IF($C7="","",SUM((Items=$C7)\*(Unique=1)\*1))

respectively.

For the final chunk, I just created two index match functions for manufacturer and category.

The functions were

=IF(D7="","",INDEX(manufacturer\_name,MATCH($C7,Part\_number,0)))

and

=IF(C7="","",INDEX(categories,MATCH($C7,Part\_number,0)))

respectively.

The final step in finishing the task was to make sure that the file calculated new results fast enough. To accomplish this I created a hard stop for addition parts using the formula:

=ROUND(SUMPRODUCT((Items<>"")/COUNTIF(Items,Items)),0).

This formula counts the amount of unique parts in Model 1 excluding blanks. I used this formula instead of a frequency function because it seemed to cause less strain on the computer. I then added an IF function in front of the additional parts formula that would return a blank if the number of rows exceeded the count of unique values. For the rest of the functions in the table, I added am IF statement that would return a blank if additional parts was blank.

This task in general went smooth. The only unforeseen issue I ran into was that some part numbers were stored as numbers but were being searched for as text. To fix this I used texts the left and Len functions to create new columns in Sales Data and Parts Data that stored all part numbers as text.