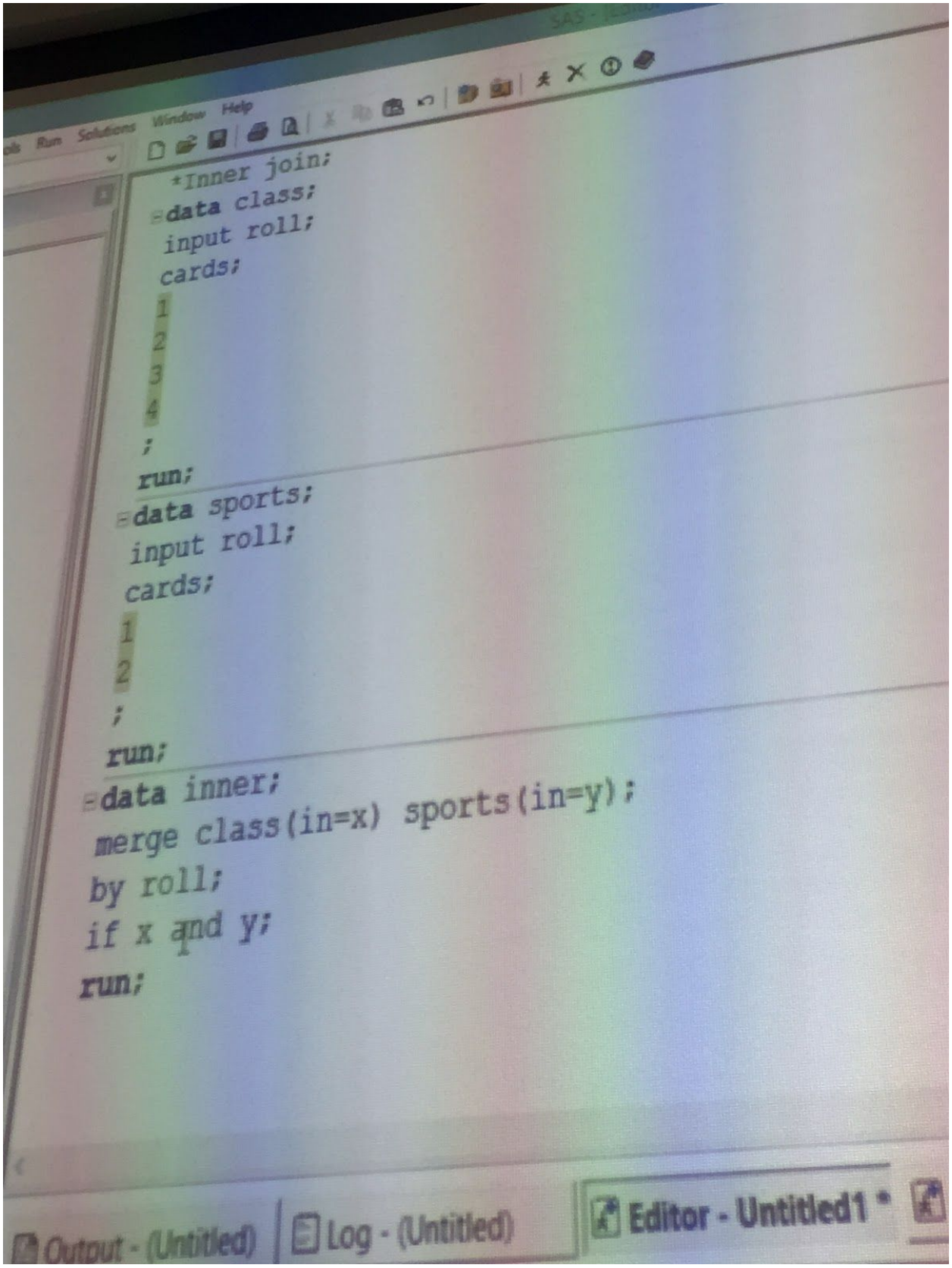
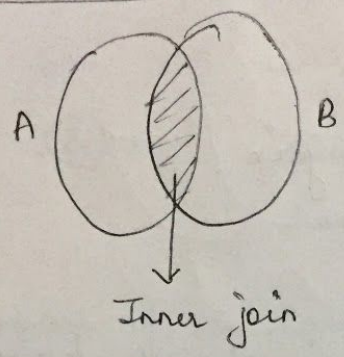
CLASS 11 NOTES



INNER JOIN




```
data class;
input roll;
cards;
1
2
3
4
5
6
;
run;
```

class	sports	x	y
1	1	1	1
2	2	1	1
3	3	1	1
4		1	0
5		1	0
6	6	1	1
	7	0	1

```
data sports;
input roll;
1
2
3
6
7
;
run;
```

```
data inner;
merge class (in=x) sports (in=y);          ——— (1)
by roll;                                   —— (2)
if x and y;  → Inner join                  —— (3)
run;
```

Explanation: In 'data class' ~~the~~ we have 6 people enrolled and in 'data sports' we have 5 people enrolled. If we want to know out of the class how many people are also enrolled in data sports, so basically we need the common people.

So, in this case we will merge data class and data sports. {(1)}

(in=x) and (in=y) are the intermediates that will create the boolean and automatic variables.

— (2) by roll; we have merged.

— (3) if x and y → inner join

{from the table see where x=1 and y=1}

"common points".

So, inner join will give the intersection of two or more datasets.

So, in base SAS we write inner join as →

"if x and y" (provided the intermediate variables have the name x and y)

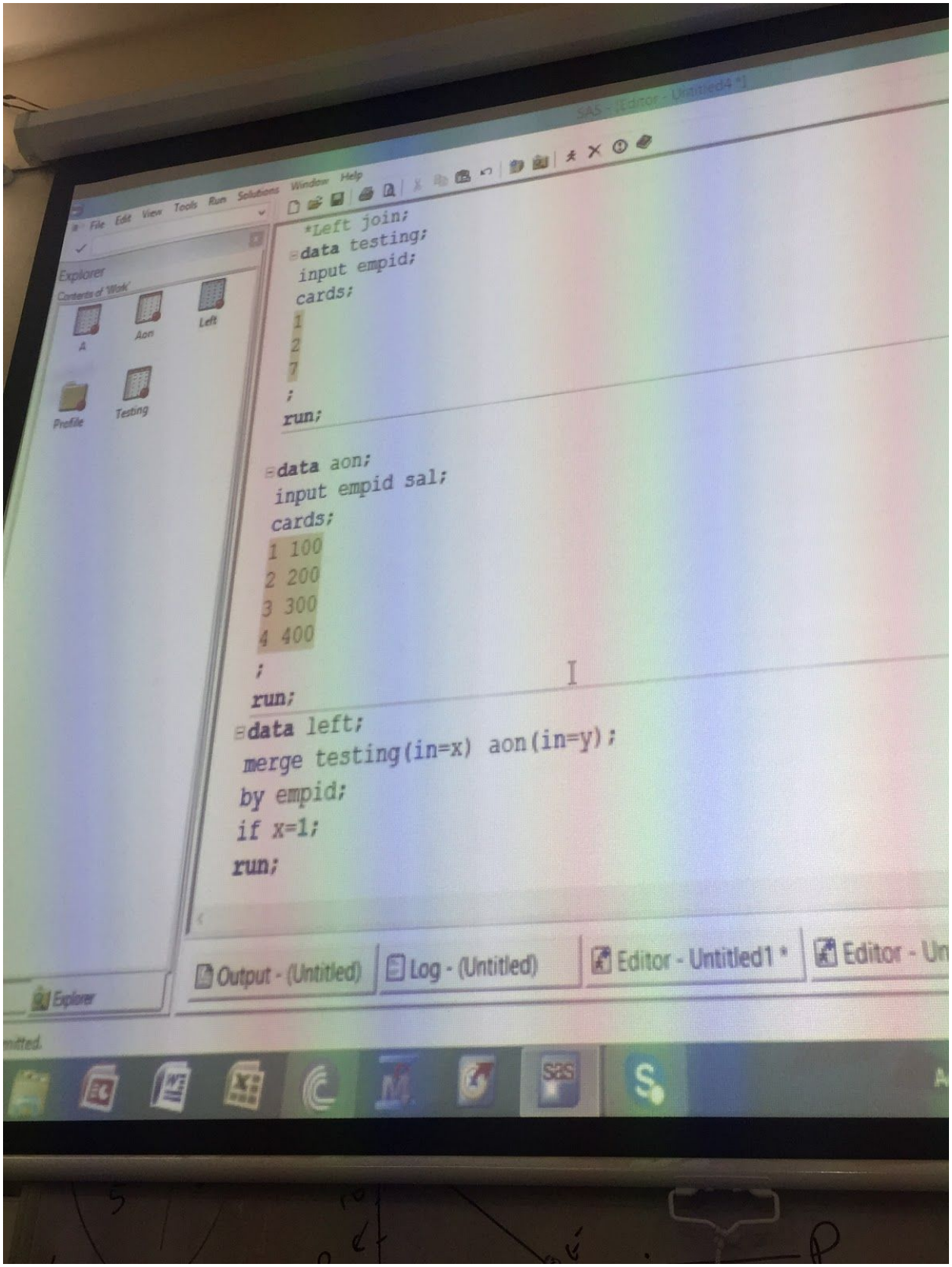
* Suppose if we have one more dataset 'z'.

then inner join — 'if x and y and z'

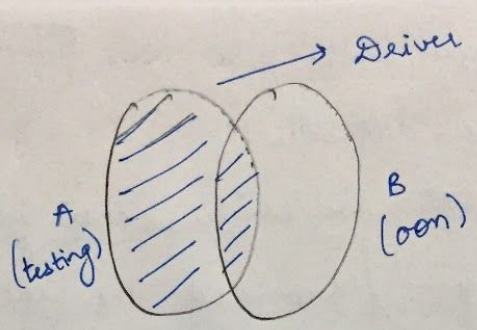
Now, O/P of program

	roll
1	1
2	2
3	3
4	6

→ intersection / common people from 'data class' and 'data sports'.


*Left join;
data testing;
input empid;
cards;
1
2
7
;
run;
data aon;
input empid sal;
cards;
1 100
2 200
3 300
4 400
;
run;
data left;
merge testing(in=x) aon(in=y);
by empid;
if x=1;
run;
Output - (Untitled)
Log - (Untitled)
Editor - Untitled1 *
Explorer
A
Aon
Left
Profile
Testing

LEFT JOIN




```
data testing;
input empid;
cards;
1
2
;
run;
```

```
data aon;
input empid sal;
cards;
1 100
2 200
3 300
4 400;
;
run;
```

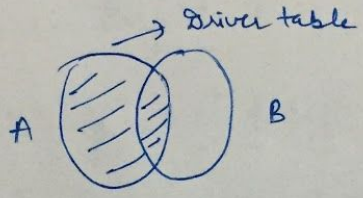
```
data left;
merge testing (in=x) aon (in=y);
by empid;
if x=1;
run;
```

→ Left join

Test	Aon	x	y
1	1	1	1
2	2	1	1
	3	0	1
	4	0	1

Suppose, we have a company 'aon' and it has a 'testing' team. (Refer to program)

In 'testing team' we have two employees and in 'aon' we have four employees and we want salary of people who are working in 'testing' team.



The table present on the left side is the 'driver table'. In figure 'A' is the driver table so everything present in 'A' will come.

And if we refer to the program 'testing' is the driver table, so everything from testing will come. irrespective of the fact whether that is present in 'aon' table or not.

So, output →

	empid	sal
1	1	100
2	2	200

→ salary of people in testing team.

Question Why left join is 'if x=1', even if we write 'if x and y' we get the same output ?
(inner join)

Take another example

```
data testing;
input empid;
cards;
1
2
7
;
run;
```

```
data aon;
input empid;
cards;
1   100
2   200
3   300
4   400
;
```

```
data left;
merge testing (in=x) aon(in=y);
by empid;
if x=1;
run;
```

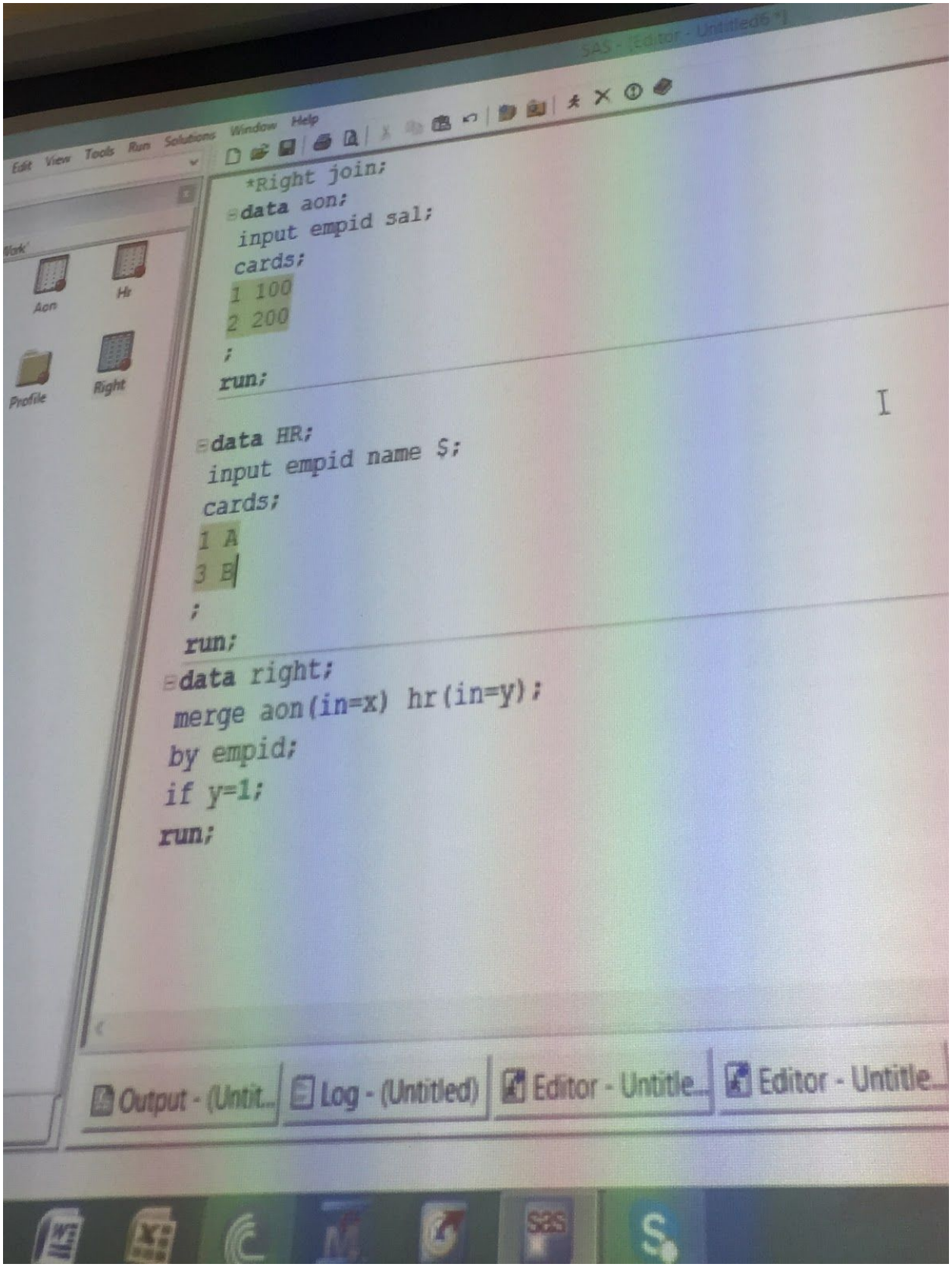
Suppose, a new person comes in a 'testing' team but its salary is not updated in the database. In this case, if we use "if x and y", then it (i.e 1 and 2) will only give the common people involved in 'testing and oon' and we will not get the information about the new person added in 'testing' team i.e. "Empid 7".

But if we write 'if x=1' { left join } then, we will be able to get the information of 'empid 7' as left join takes all the values present in the left (driver) table. So, the output will be:

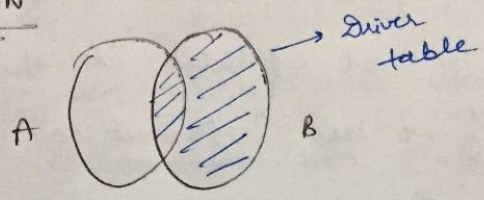
	empid	sal
1	1	100
2	2	200
3	7	.

Interview Ques: When would left join and inner join gives the same output?

- When the table on the left side is a 'subset' of table on the right side.


Edit View Tools Run Solutions Window Help
Aon
Hr
Profile
Right
*Right join;
data aon;
input empid sal;
cards;
1 100
2 200
;
run;
data HR;
input empid name $;
cards;
1 A
3 B
;
run;
data right;
merge aon(in=x) hr(in=y);
by empid;
if y=1;
run;
Output - (Untit...
Log - (Untitled)
Editor - Untitle...
Editor - Untitle...

RIGHT JOIN




```
data aon;
input empid sal;
1   100
2   200
;
run;
```

```
data HR;
input empid name $;
cards;
1  A
3  B
;
run;
```

```
data right;
merge aon (in=x) hr (in=y);
by emp id;
if y=1;
run;
```

→ Right join

aon	hr	x	y
1	1	1	1
2		1	0
	3	0	1

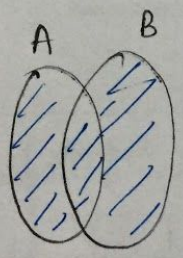
Now, here 'HR' is the driver table (right table) so everything from 'hr' will come irrespective of the data present in 'oon' table.

So, output →

	empid	sal	name
1	1	100	A
2	3	.	B

(Data from right (driver) table)

FULL JOIN

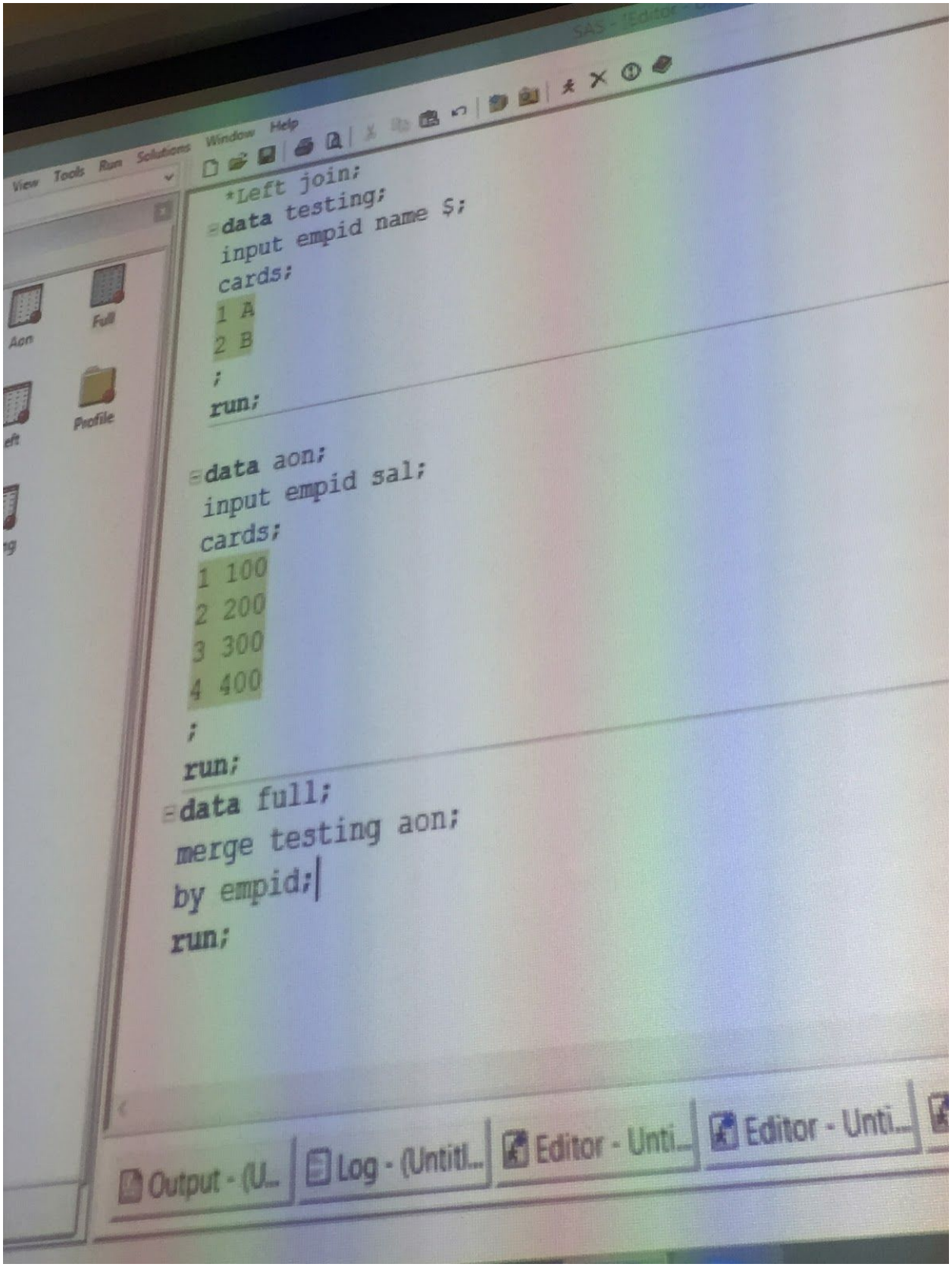


The default merge we do is full join.

Refer to previous prog:

```
data full;
merge oon hr;
by emp id;
run;
```

	emp id	sal	name
1	1	100	A
2	2	200	
3	3	.	B


*Left join;
data testing;
input empid name $;
cards;
1 A
2 B
;
run;
data aon;
input empid sal;
cards;
1 100
2 200
3 300
4 400
;
run;
data full;
merge testing aon;
by empid;
run;
Output - (U...
Log - (Untitl...
Editor - Unti...
Editor - Unti...

CASE STUDY - 4

```
data testing;
input empid name $;
cards;
1   A
2   B
;
run;
```

```
data aon;
input id sal;
cards;
1   100
2   200
;
run;
```

Different input key.

```
data full;
merge testing aon (rename = (id = empid));
by empid;
run;
```

↓

(dataset option)

As per the condition of merging in base SAS, we should have the common variable name only then we can do merging.

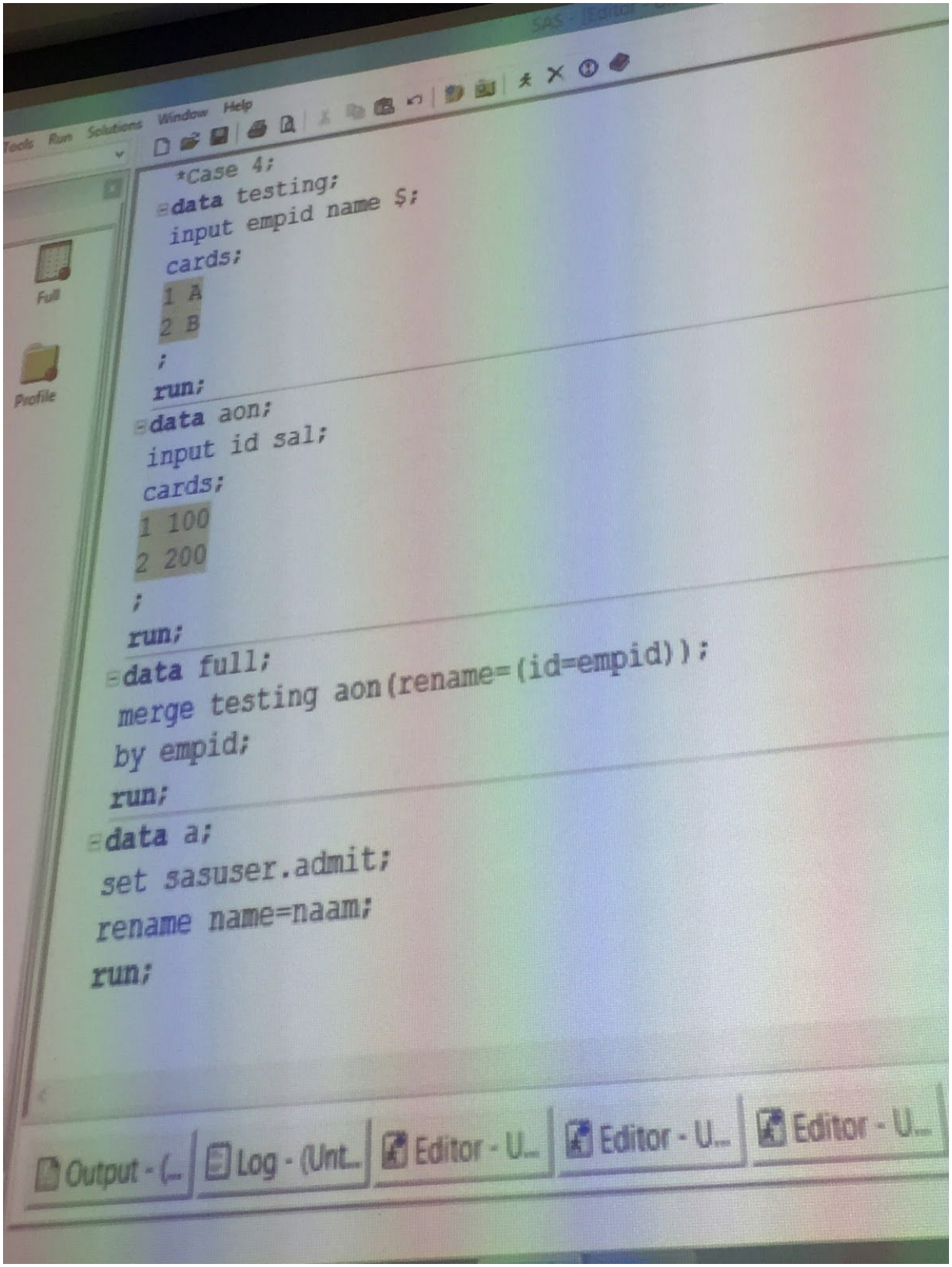
Here in 'data testing' we have input key 'empid' and in 'data aon' we have input key 'id' which do not match.

So, in order to merge 'testing and aon' we will change the variable name of 'id' and will rename it as 'empid' so that the requirement of merging is fulfilled.

Note : Here 'rename' is dataset option.

	empid	name	sal
1	1	A	100
2	2	B	200

We have used rename as dataset option in the program.


Tools Run Solutions Window Help
Full
Profile
*Case 4;
data testing;
input empid name $;
cards;
1 A
2 B
;
run;
data aon;
input id sal;
cards;
1 100
2 200
;
run;
data full;
merge testing aon(rename=(id=empid));
by empid;
run;
data a;
set sasuser.admit;
rename name=naam;
run;
Output - (...
Log - (Unt...
Editor - U...
Editor - U...
Editor - U...

"Using rename as a statement"

```
data testing ;
input empid name $
cards;
1  A
2  B
;
run;
```

```
data aon;
input id sal;
cards;
1  100
2  200
;
run;
```

```
data full;
merge testing aon (rename=(id = empid));
by empid;
run;
```

→ (rename as dataset option)

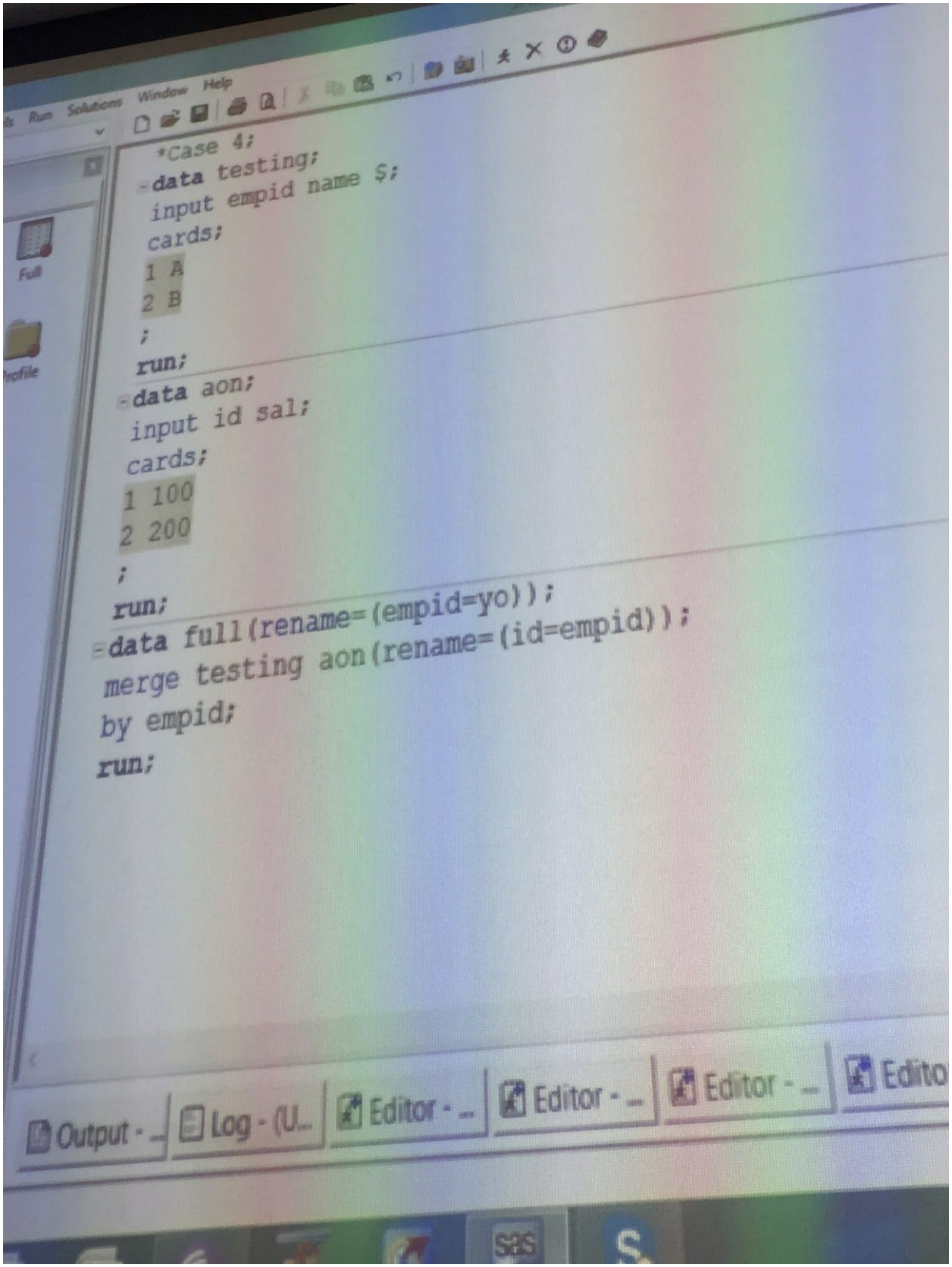
```
data a;
set sasuser.admit;
rename name = naam;
run;
```

→ (rename as statement)

changed/renamed from name to naam

	empid	naam	sal
1	1	A	100
2	2	B	200

OUTPUT


Run Solutions Window Help
*Case 4;
data testing;
input empid name $;
cards;
1 A
2 B
;
run;
data aon;
input id sal;
cards;
1 100
2 200
;
run;
data full(rename=(empid=yo));
merge testing aon(rename=(id=empid));
by empid;
run;
Output - _
Log - (U...
Editor - _
Editor - _
Editor - _
Edito

We can also write from previous prog:

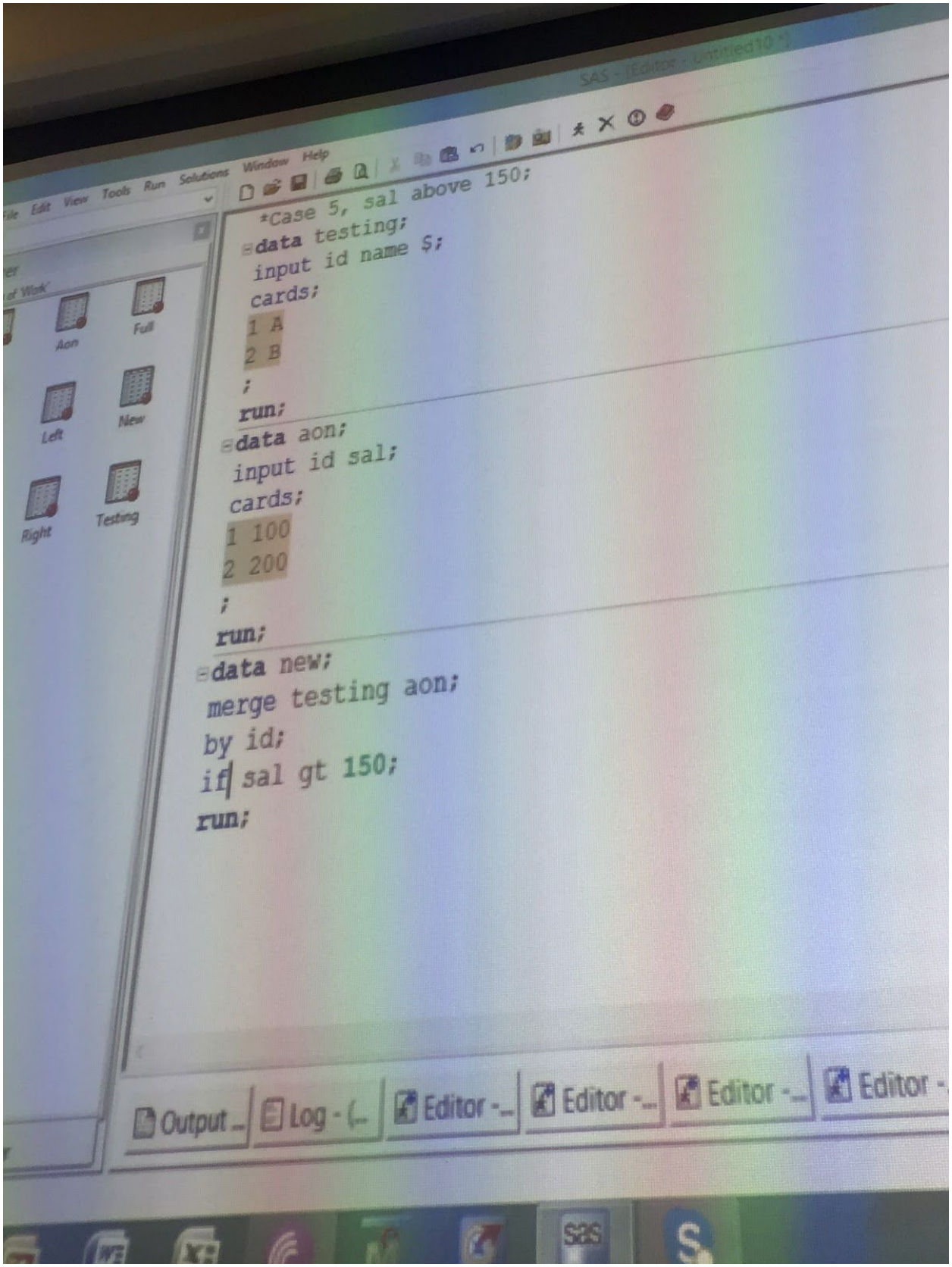
```
data full (rename = (empid = yo));        — (1)
merge testing aon (rename = (id = empid));
by empid;
run;
```

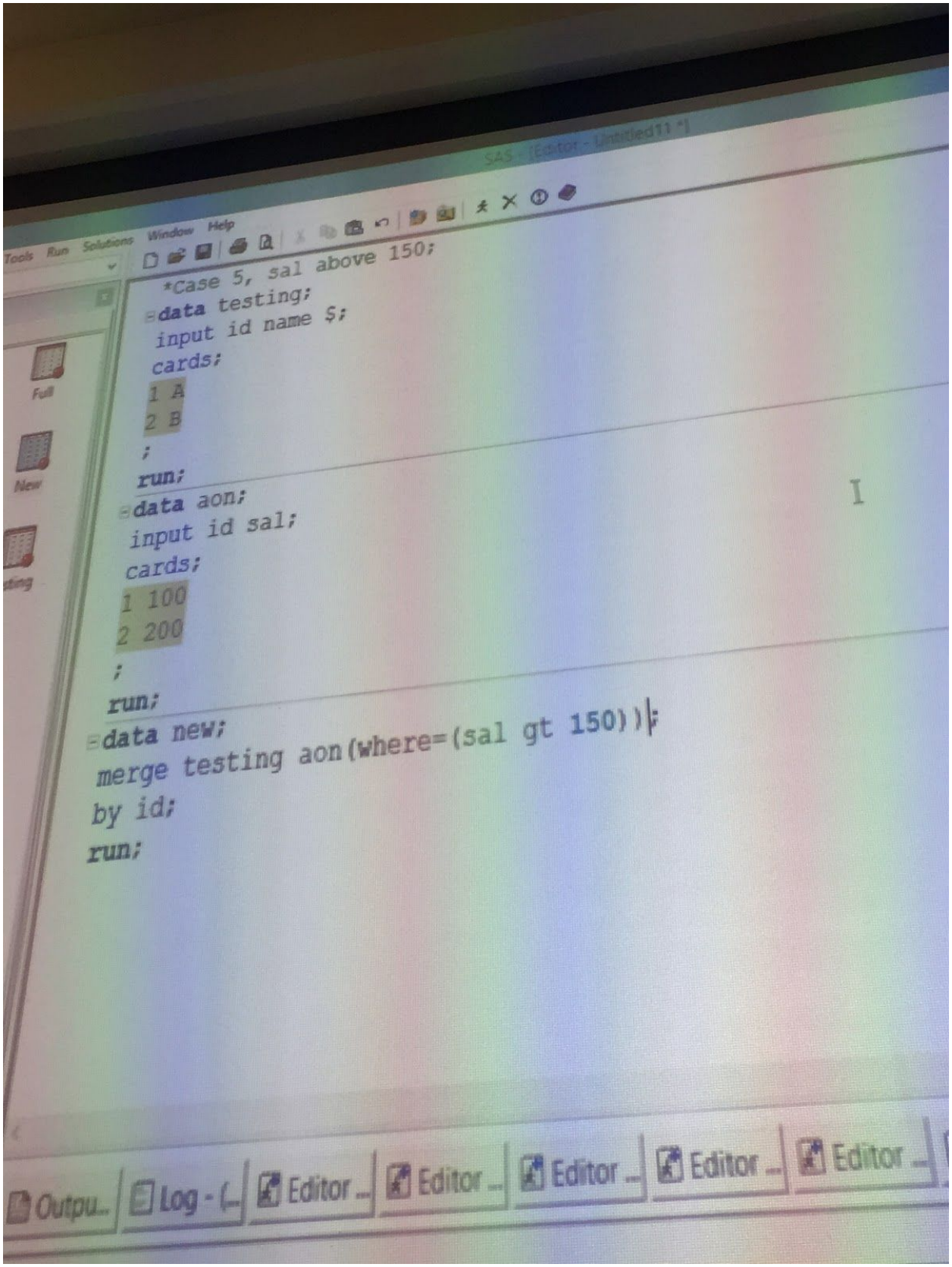
during merging 'id' name will change to 'empid'

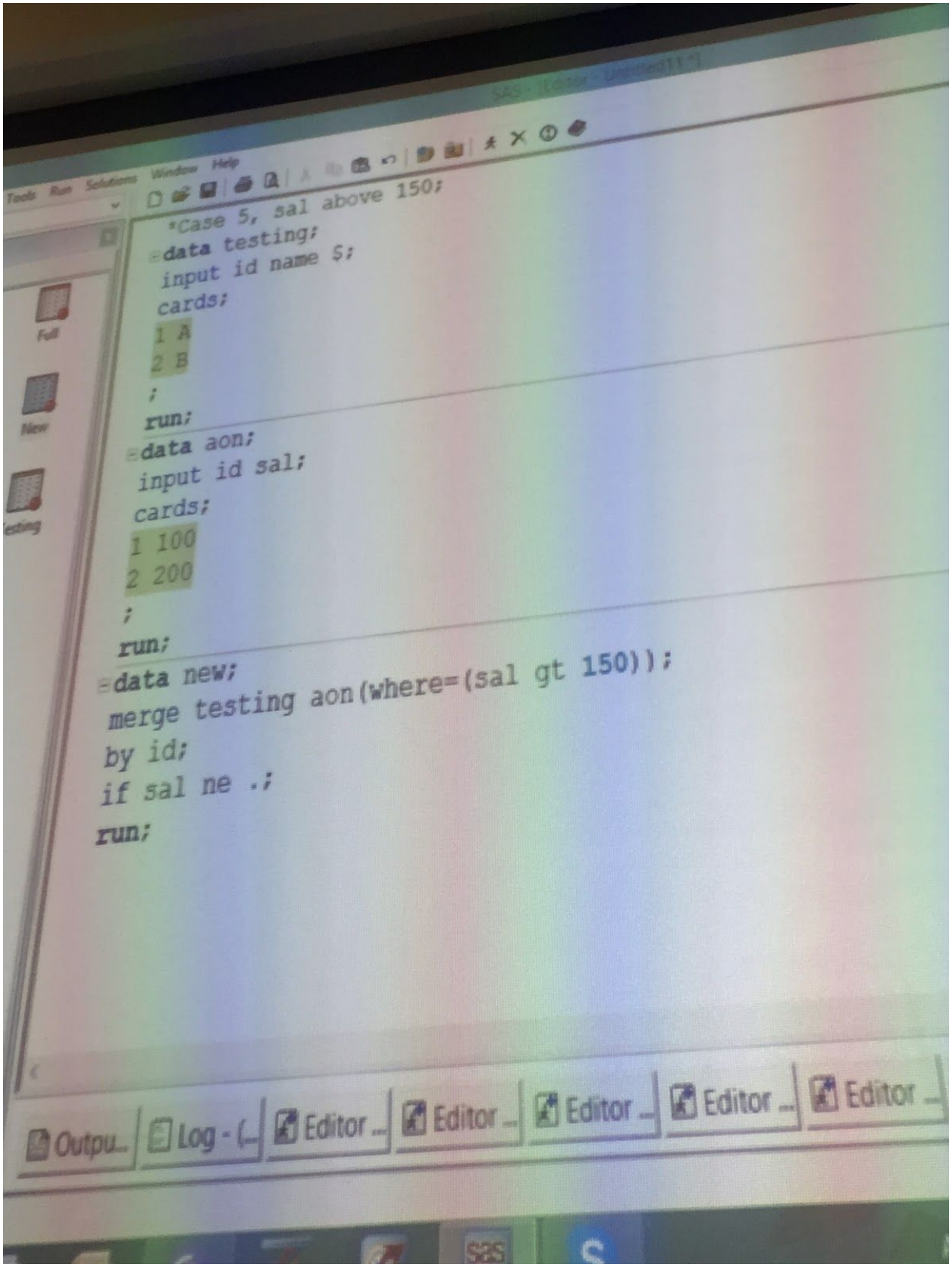
and in (1) while writing the data name of 'empid' will change to 'yo'.

	yo	name	sal
1	1	A	100
2	2	B	200

OUTPUT


Edit View Tools Run Solutions Window Help
Aon
Full
Left
New
Right
Testing
*Case 5, sal above 150;
data testing;
input id name $;
cards;
1 A
2 B
;
run;
data aon;
input id sal;
cards;
1 100
2 200
;
run;
data new;
merge testing aon;
by id;
if sal gt 150;
run;
Output
Log - (
Editor -
Editor -
Editor -
Editor -


Tools Run Solutions Window Help
*Case 5, sal above 150;
data testing;
input id name $;
cards;
1 A
2 B
;
run;
data aon;
input id sal;
cards;
1 100
2 200
;
run;
data new;
merge testing aon(where=(sal gt 150));
by id;
run;
Full
New
Outpu...
Log - (...
Editor ...
Editor ...
Editor ...
Editor ...
Editor ...


*Case 5, sal above 150;
data testing;
input id name $;
cards;
1 A
2 B
;
run;
data aon;
input id sal;
cards;
1 100
2 200
;
run;
data new;
merge testing aon(where=(sal gt 150));
by id;
if sal ne .;
run;
Outpu...
Log - (...
Editor ...
Editor ...
Editor ...
Editor ...
Editor ...

CASE STUDY - 5 , sal above 150;

```
data testing;
input id name $;
cards;
1 A
2 B
;
run;
```

```
data aon;
input id sal;
cards;
1 100
2 200
;
run;
```

— ①

```
data new;
merge testing aon;
by id
where sal gt 150;
run;
```

— ②

```
data new;
merge testing aon;
by id;
if sal gt 150;
run;
```

Here, we want the employees whose salary are above 150.

If we refer to prog ① { where sal gt 150; }

This code will show error because "where" is applied on parent dataset. for new, parents are testing and aon, so "where" will search the 'sal' variable in both testing and 'aon' but the 'sal' variable is present in aon and not in testing so, will show error.

If we want to use "where" then the 'sal' variable should be present in both aon and testing, only then it will give the desired result.

So, in this case we will use 'if' { prog ② }

(if sal gt 150;).

Here, 'if' will allow the merge to happen and then it will apply filter and will give the desired result.

	id	name	sal
1	2	B	200

→ output

'where' as a data set option

```
data testing ;
input id name$;
cards;
1  A
2  B
;
run;
```

```
data aon;
input id sal;
cards;
1  100
2  200
;
run;
```

If we want to apply filter with 'where' only then use 'where' as a dataset option with aon and this will give the result.

```
data new;
merge testing aon (where = (sal gt 150));
by id ;
run;
```

↘ data set option

Output

	id	name	sal
1	1	A	.
2	2	B	200

But the desired result we wants is →

	id	name	sal
1	2	B 1	200

then, in this case write the prog as:

```
data new;
merge testing aon (where = (sal gt 150));
by id;
if sal ne . ;
run;
```

→ This will give the above result.

Note : Till now we have used four data set options:

1. Keep
2. Drop
3. Rename
4. Where

```
data testing;
input id name;
cards;
1 A
2 B
;
run;
```

```
data aon;
input id sal;
cards;
10 100
22 200
;
run;
```

```
data new;
merge testing ion;
by id;
run;
```

```
data new;
merge testing aon;
run;
```

} Another case of }
Not using 'by'

(1) → simple merging is done.

(2) → here 'by' is not used.

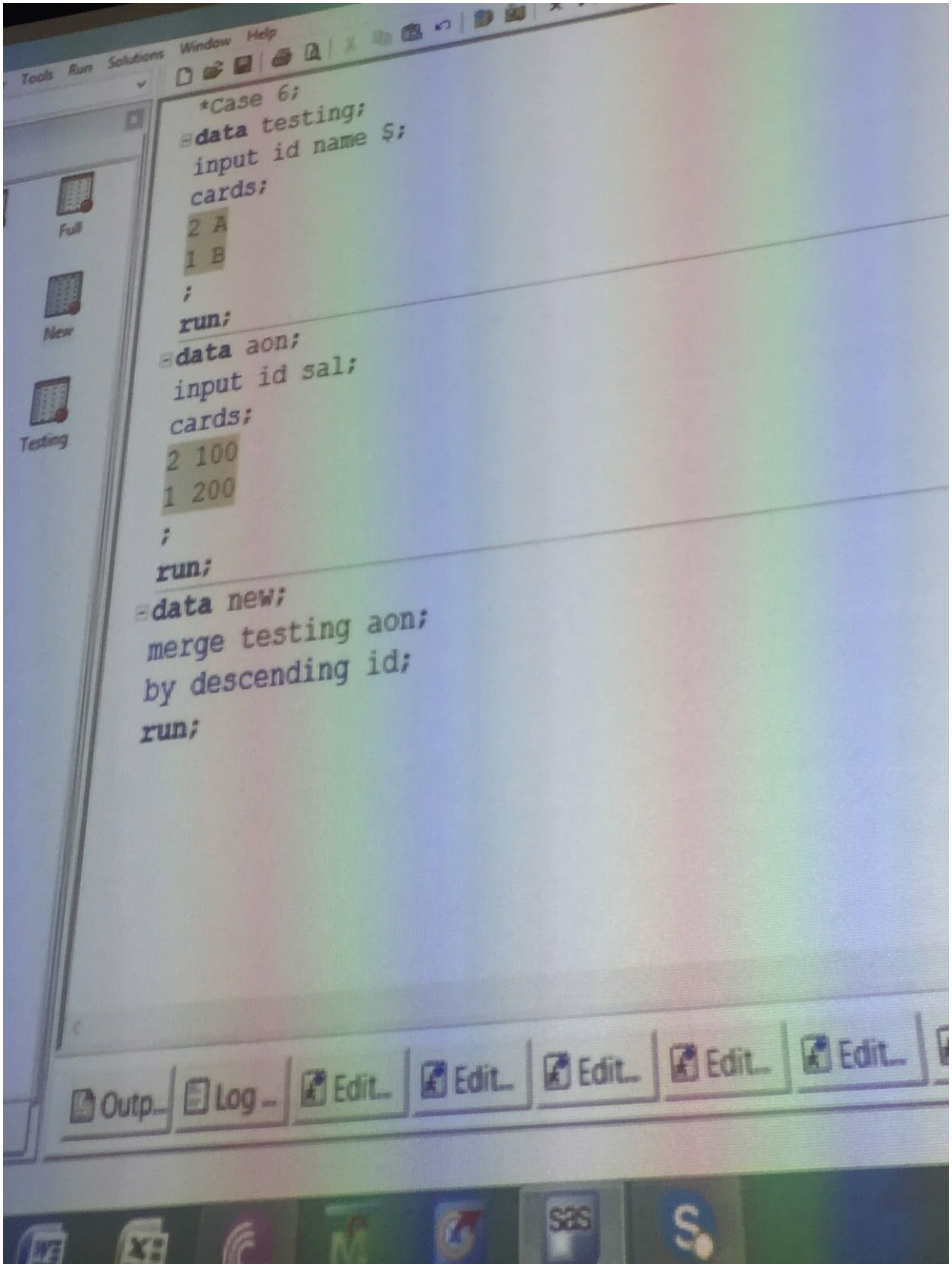
If we do not use 'by' then the 2nd dataset will simply overwrite the 1st one irrespective of looking for any common key.

Output → from (2)

	id	name	sal
1	10	A	100
2	22	B	200

from (1)

	id	name	sal
1	1	A	.
2	2	B	.
3	10		100
4	22		200


Tools Run Solutions Window Help
Full
New
Testing
*Case 6;
data testing;
input id name $;
cards;
2 A
1 B
;
run;
data aon;
input id sal;
cards;
2 100
1 200
;
run;
data new;
merge testing aon;
by descending id;
run;
Outp...
Log...
Edit...
Edit...
Edit...
Edit...
Edit...

CASE STUDY - 6

```
data testing ;
input id name$;
cards;
2   A
1   B
;
run;
```

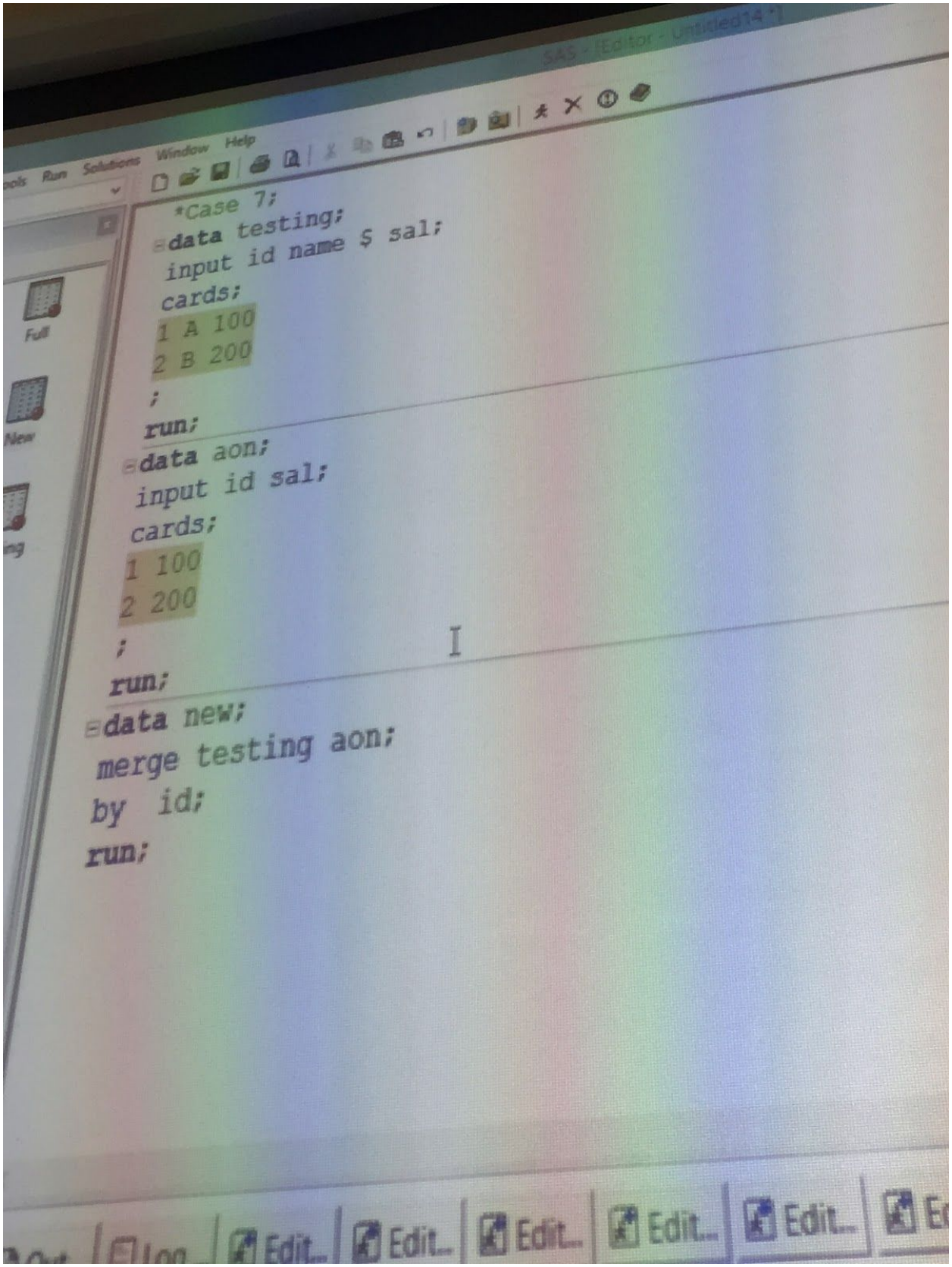
```
data aon;
input id sal;
cards;
2   100
1   200
;
run;
```

```
data new;
merge testing aon;
by descending id;
run;
```

Here, id's in both the datasets are in descending order. No need to arrange them in ascending order. Just use descending sorting to get the desired result.

Output

	id	name	sal
1	2	A	100
2	1	B	200


Run Solutions Window Help
*Case 7;
data testing;
input id name $ sal;
cards;
1 A 100
2 B 200
;
run;
data aon;
input id sal;
cards;
1 100
2 200
;
run;
data new;
merge testing aon;
by id;
run;
Edit...
Edit...
Edit...
Edit...
Edit...

CASE STUDY - 7

```
data testing;
input id name $ sal;
cards;
1  A  100
2  B  200
;
;
run;
```

```
data aon;
input id sal;
cards;
1  100
2  200
;
;
```

```
data new;
merge testing aon;
by id;
run;
```

we do merging if we want to consolidate data. Here, we do not require merging as we have the common variable with same values in both the datasets testing and aon.
Here dataset aon is a complete subset of dataset testing, so no merging required.