

12.1

The screenshot shows the SQL Developer interface with a script titled "SQL File 3". The script contains the following SQL code:

```
1 create or replace view open_items
2 as
3 select vendor_name, invoice_number, invoice_total,
4        invoice_total - payment_total - credit_total as balance_due
5 from vendors join invoices
6      on vendors.vendor_id = invoices.vendor_id
7 where invoice_total - payment_total - credit_total > 0
8 order by vendor_name;
```

The "Output" pane at the bottom shows the execution results:

#	Time	Action	Message
1	15:19:24	create or replace view open_items as select vendor_name, invoice_number, invoice_total, invoice_total - payment_total - credit_total as balance_due from vendors join invoices on vendors.vendor_id = invoices.vendor_id where invoice_total - payment_total - credit_total > 0 order by vendor_name;	0 row(s) affected

12.2

The screenshot shows the SQL Developer interface with a script titled "SQL File 3". The script contains the following SQL code:

```
6      on vendors.vendor_id = invoices.vendor_id
7 where invoice_total - payment_total - credit_total > 0
8 order by vendor_name;
9
10 select *
11 from open_items
12 where balance_due >= 1000;
```

The "Result Grid" pane at the bottom shows the execution results:

vendor_name	invoice_number	invoice_total	balance_due
Mallov Lithorachino Inc	P-0608	20551.18	19351.18
Mallov Lithorachino Inc	O-2436	10976.06	10976.06

12.3

The screenshot shows the SQL Developer interface with a script titled "SQL File 3". The script contains the following SQL code:

```
9
10 select *
11 from open_items
12 where balance_due >= 1000;
13
14 create or replace view open_items_summary
15 as
16 select vendor_name, count(*) as open_item_count,
17        sum(invoice_total - credit_total - payment_total) as open_item_total
18 from vendors join invoices
19      on vendors.vendor_id = invoices.vendor_id
20 where invoice_total - credit_total - payment_total > 0
21 group by vendor_name
22 order by open_item_total desc;
```

The "Output" pane at the bottom shows the execution results:

#	Time	Action	Message	Duration / Fetch
1	15:29:16	create or replace view open_items_summary as select vendor_name, count(*) as open_item_count, sum(invoice_total - credit_total - payment_total) as open_item_total from vendors join invoices on vendors.vendor_id = invoices.vendor_id where invoice_total - credit_total - payment_total > 0 group by vendor_name order by open_item_total desc;	0 row(s) affected	0.016 sec

12.4

SQL File 3

```
20 where invoice_total - credit_total - payment_total > 0
21 group by vendor_name
22 order by open_item_total desc;
23
24 select *
25 from open_items_summary
26 limit 5;
27
28
29
```

Result Grid

vendor_name	open_item_count	open_item_total
Mallory LithoGraphics Inc	2	30327.24
Inoram	1	579.42

open_items_summary 2

Read Only

12.5

SQL File 3

```
18 from vendors join invoices
19 on vendors.vendor_id = invoices.vendor_id
20 where invoice_total - credit_total - payment_total > 0
21 group by vendor_name
22 order by open_item_total desc;
23
24 select *
25 from open_items_summary
26 limit 5;
27
28 create or replace view vendor_address
29 as
30 select vendor_id, vendor_address1, vendor_address2, vendor_city, vendor_state, vendor_zip_code
31 from vendors;
32
33
34
```

12.6

SQL File 3

```
21 group by vendor_name
22 order by open_item_total desc;
23
24 select *
25 from open_items_summary
26 limit 5;
27
28 create or replace view vendor_address
29 as
30 select vendor_id, vendor_address1, vendor_address2, vendor_city, vendor_state, vendor_zip_code
31 from vendors;
32
33 update vendor_address
34 set vendor_address1 = '1990 Westwood Blvd',
35 vendor_address2 = 'Ste 260'
36 where vendor_id = 4;
37
```

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Context Help Snippets

Output

#	Time	Action	Message	Duration / Fetch
1	15:34:57	update vendor_address set vendor_address1 = '1990 Westwood Blvd', vendor_address2 = 'Ste 260' where vendor_id = 4;	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0	0.015 sec

13.1

The first screenshot shows the initial SQL script in SQL Server Enterprise Manager. The script is as follows:

```
1 use ap;
2
3 drop procedure if exists test;
4
5 -- Change statement delimiter from semicolon to double front slash
6 delimiter //
7
8 create procedure test()
9 begin
10 declare invoice_count int;
11
12 select count(*)
13 into invoice_count
14 from invoices
15 where invoice_total - payment_total - credit_total >= 5000;
16
17 select concat(invoice_count, ' invoices exceed $5000.') as message;
18 end//
```

The second screenshot shows the same script with the addition of a call to the procedure at the end:

```
19
20 -- Change statement delimiter from semicolon to double front slash
21 DELIMITER ;
22
23 call test();
```

The third screenshot shows the execution results in the 'Result Grid' pane. The results are as follows:

message
2 invoices exceed \$5000.

The bottom of the third screenshot shows the 'Result 1' tab and a 'Read Only' status indicator.

13.2

The screenshot shows the SQL script for creating a stored procedure named test(). The script is as follows:

```
1 use ap;
2
3 drop procedure if exists test;
4
5 -- change statement delimiter from semicolon to double front slash
6 delimiter //
7
8 create procedure test()
9 begin
10 declare count_balance_due int;
11 declare total_balance_due decimal(9,2);
12
13 select count(*), sum(invoice_total - payment_total - credit_total)
14 into count_balance_due, total_balance_due
15 from invoices
16 where invoice_total - payment_total - credit_total > 0;
17
18 if total_balance_due > 50000 then
```

The screenshot shows a SQL query in a script editor. The query calculates the total balance due for invoices where the total (invoice_total minus payment_total minus credit_total) is greater than zero. It uses a conditional select to format the output based on whether the total balance due is greater than or equal to 30,000. The results grid shows one row with a count of 11 and a total balance due of 32020.42.

```
14 into count_balance_due, total_balance_due
15 from invoices
16 where invoice_total - payment_total - credit_total > 0;
17
18 if total_balance_due >= 30000 then
19     select count_balance_due as count_balance_due,
20            total_balance_due as total_balance_due;
21 else
22     select 'total balance due is less than $30,000.' as message;
23 end if;
24 end//
25
26 -- change statement delimiter from semicolon to double front slash
27 delimiter ;
28
29 call test();
30
```

Result Grid

count_balance_due	total_balance_due
11	32020.42

Result 2 x Read Only

13.3

The screenshot shows a SQL procedure named test() in a script editor. The procedure declares two integer variables, i and factorial, both with default values of 10. It then enters a while loop that calculates the factorial of i, decrementing i by 1 until it reaches 1. The results grid shows a message: 'The factorial of 10 is: 3.628.800.'

```
1 drop procedure if exists test;
2
3 delimiter //
4
5 create procedure test()
6 begin
7     declare i int default 10;
8     declare factorial int default 10;
9
10    while i > 1 DO
11        set factorial = factorial * (i - 1);
12    end while;
13
14    select concat('The factorial of 10 is: ', format(factorial,0), '.') as message;
15 end//
```

Result Grid

message
The factorial of 10 is: 3.628.800.

Result 2 x Read Only

SQL File 3* SQL File 10* x

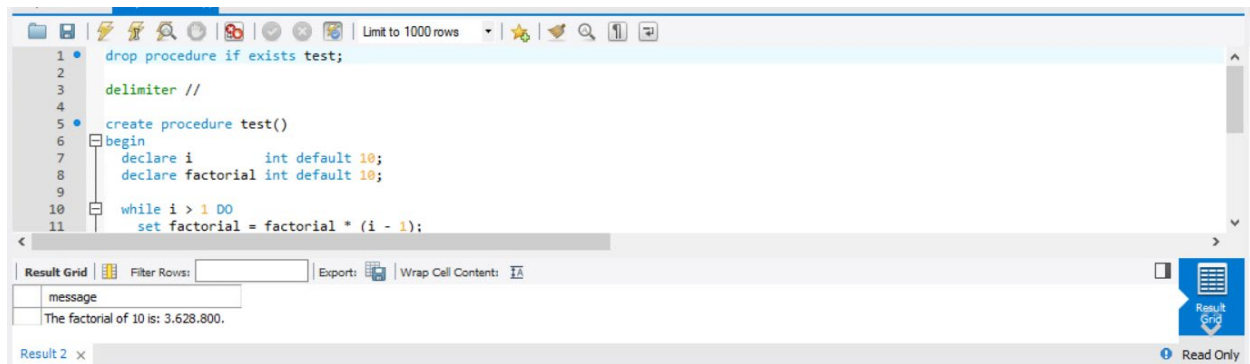
```
7 declare i int default 10;
8 declare factorial int default 10;
9
10 while i > 1 DO
11     set factorial = factorial * (i - 1);
12     set i = i - 1;
13 end while;
14
15 select concat('The factorial of 10 is: ', format(factorial,0), '.') as message;
16
17 end//
```

Result Grid

message
The factorial of 10 is: 3.628.800.

Result 2 x Read Only

13.3



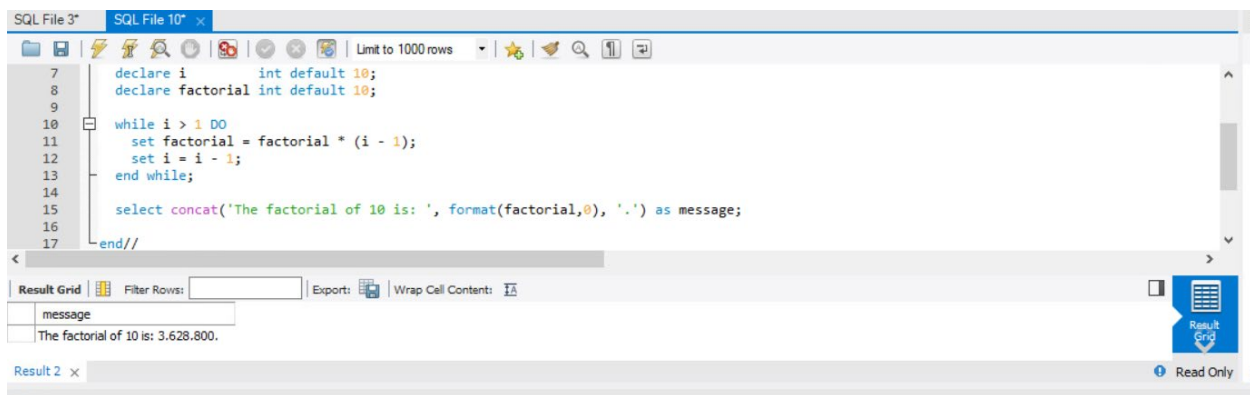
The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the code for a stored procedure named 'test'. The code starts with 'drop procedure if exists test;', followed by 'delimiter //' and 'create procedure test()'. Inside the procedure, there is a 'begin' block containing 'declare i int default 10;', 'declare factorial int default 10;', a 'while i > 1 DO' loop, and 'set factorial = factorial * (i - 1);'. The bottom pane shows the 'Result Grid' with a single row containing the message 'The factorial of 10 is: 3.628.800.'

```
1 drop procedure if exists test;
2
3 delimiter //
4
5 create procedure test()
6 begin
7     declare i int default 10;
8     declare factorial int default 10;
9
10    while i > 1 DO
11        set factorial = factorial * (i - 1);
12    end while;
13 end;
```

Result Grid

message
The factorial of 10 is: 3.628.800.

Result 2 x Read Only



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the full code for the stored procedure 'test'. The code includes 'declare i int default 10;', 'declare factorial int default 10;', a 'while i > 1 DO' loop, and 'set factorial = factorial * (i - 1);'. The bottom pane shows the 'Result Grid' with a single row containing the message 'The factorial of 10 is: 3.628.800.'

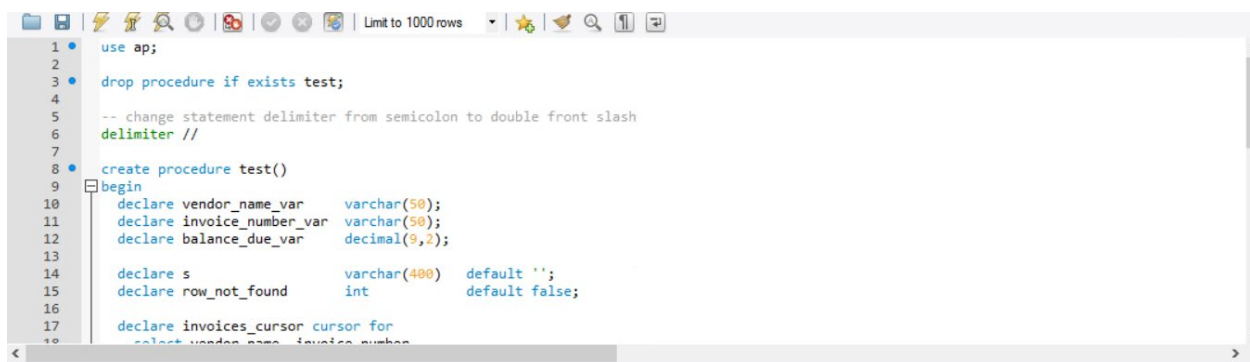
```
7 declare i int default 10;
8 declare factorial int default 10;
9
10 while i > 1 DO
11     set factorial = factorial * (i - 1);
12     set i = i - 1;
13 end while;
14
15 select concat('The factorial of 10 is: ', format(factorial,0), '.') as message;
16
17 end//
```

Result Grid

message
The factorial of 10 is: 3.628.800.

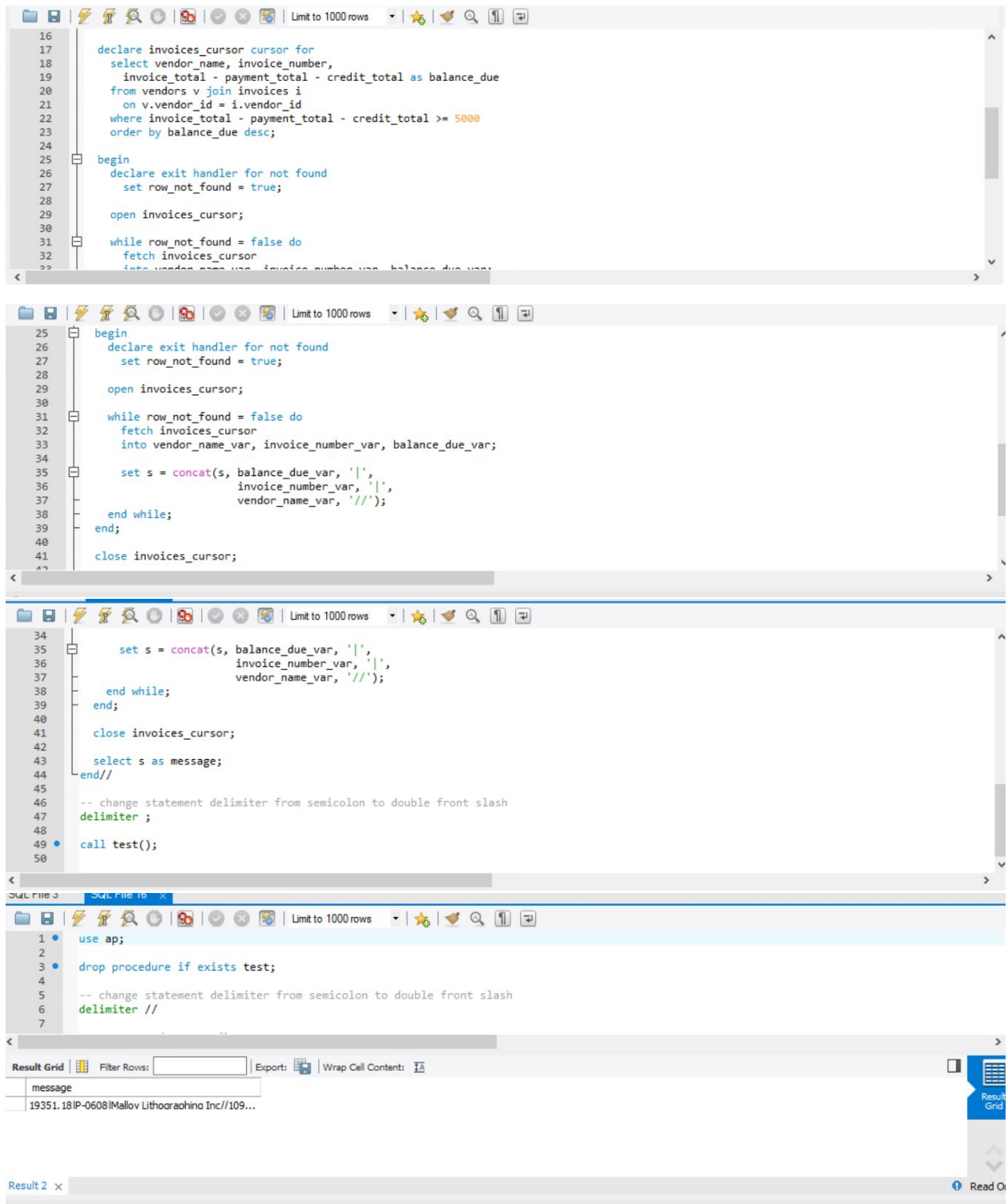
Result 2 x Read Only

13.4



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the code for a stored procedure named 'test'. The code starts with 'use ap;', followed by 'drop procedure if exists test;', and 'delimiter //'. Inside the procedure, there is a 'begin' block containing 'declare vendor_name_var varchar(50);', 'declare invoice_number_var varchar(50);', 'declare balance_due_var decimal(9,2);', 'declare s varchar(400) default '';', 'declare row_not_found int default false;', and 'declare invoices_cursor cursor for select vendor_name, invoice number'.

```
1 use ap;
2
3 drop procedure if exists test;
4
5 -- change statement delimiter from semicolon to double front slash
6 delimiter //
7
8 create procedure test()
9 begin
10     declare vendor_name_var varchar(50);
11     declare invoice_number_var varchar(50);
12     declare balance_due_var decimal(9,2);
13
14     declare s varchar(400) default '';
15     declare row_not_found int default false;
16
17     declare invoices_cursor cursor for
18         select vendor_name, invoice number
```



```

16
17 declare invoices_cursor cursor for
18   select vendor_name, invoice_number,
19     invoice_total - payment_total - credit_total as balance_due
20   from vendors v join invoices i
21     on v.vendor_id = i.vendor_id
22   where invoice_total - payment_total - credit_total >= 5000
23   order by balance_due desc;
24
25 begin
26   declare exit handler for not found
27     set row_not_found = true;
28
29   open invoices_cursor;
30
31   while row_not_found = false do
32     fetch invoices_cursor
33     into vendor_name_var, invoice_number_var, balance_due_var;
34
35     set s = concat(s, balance_due_var, '|',
36                   invoice_number_var, '|',
37                   vendor_name_var, '///');
38
39   end while;
40 end;
41 close invoices_cursor;
42
43 select s as message;
44 end//
45
46 -- change statement delimiter from semicolon to double front slash
47 delimiter ;
48
49 call test();
50

```

Result Grid

message
19351.18 IP-0608 Mollov Lithooraahina Inc//109...

Result 2 x

Read O

The screenshot displays three panes in SQL Studio:

- Top Pane (SQL Editor):** Contains SQL code for updating invoices and a test procedure.


```

13
14 update invoices
15 set invoice_due_date = null
16 where invoice_number = '989319-457';
17
18 if column_cannot_be_null = true then
19   select 'row was not updated - column cannot be null.' as message;
20 else
21   select '1 row was updated.' as message;
22 end if;
23
24 end//
25
26 delimiter ;
27
28 • call test();
29
      
```
- Middle Pane (Result Grid):** Shows the output of the test procedure.

message
row was not updated - column cannot be null.
- Bottom Pane (Source Editor):** Shows the full source code of the test procedure.


```

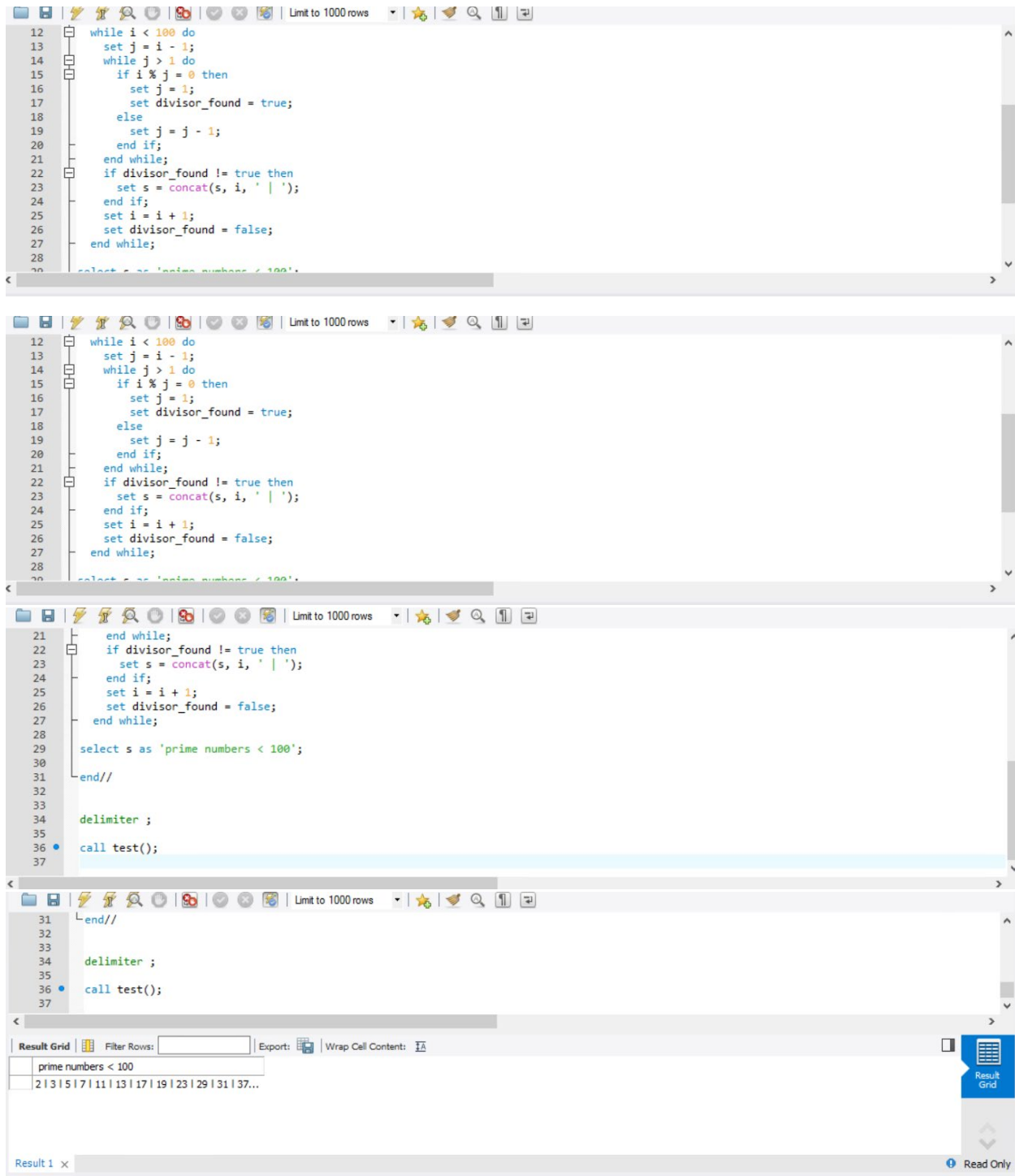
1 use ap;
2
3 • drop procedure if exists test;
4
5 delimiter //
6
7 • create procedure test()
8 begin
9   declare column_cannot_be_null tinyint default false;
10
11   declare continue handler for 1048
12     set column_cannot_be_null = true;
13
14   update invoices
15   set invoice_due_date = null
16   where invoice_number = '989319-457';
17
18   if column_cannot_be_null = true then
      
```

13.6

The screenshot displays the source code of a test procedure in SQL Studio:

```

1 drop procedure if exists test;
2
3 delimiter //
4
5 • create procedure test()
6 begin
7   declare i int default 1;
8   declare j int;
9   declare divisor_found tinyint default true;
10  declare s varchar(400) default '';
11
12  while i < 100 do
13    set j = i - 1;
14    while j > 1 do
15      if i % j = 0 then
16        set j = 1;
17        set divisor_found = true;
18      else
      
```



```
12 while i < 100 do
13   set j = i - 1;
14   while j > 1 do
15     if i % j = 0 then
16       set j = 1;
17       set divisor_found = true;
18     else
19       set j = j - 1;
20     end if;
21   end while;
22   if divisor_found != true then
23     set s = concat(s, i, ' ');
24   end if;
25   set i = i + 1;
26   set divisor_found = false;
27 end while;
28 select s as 'prime numbers < 100';
29
30
31 end//
32
33 delimiter ;
34
35 call test();
36
37
```

Result Grid

prime numbers < 100
2 3 5 7 11 13 17 19 23 29 31 37...

Result 1 x Read Only


```

1  use ap;
2
3  drop procedure if exists test;
4
5  -- change statement delimiter from semicolon to double front slash
6  delimiter //
7
8  create procedure test()
9  begin
10 declare vendor_name_var    varchar(50);
11 declare invoice_number_var  varchar(50);
12 declare balance_due_var     decimal(9,2);
13
14 declare s                   varchar(400) default '';
15 declare row_not_found      int          default false;
16
17 declare invoices_cursor cursor for
18   select vendor_name, invoice_number,
19     invoice_total - payment_total - credit_total as balance_due
20   from vendors v join invoices i
21     on v.vendor_id = i.vendor_id
22   where invoice_total - payment_total - credit_total >= 5000
23   order by balance_due desc;
24
25 -- loop 1
26 begin
27   declare exit handler for not found
28     set row_not_found = true;
29
30   open invoices_cursor;
31
32   set s = concat(s, '$20,000 or more: ');
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49 set row_not_found = false;
50 begin
51   declare exit handler for not found
52     set row_not_found = true;
53
54   open invoices_cursor;
55
56   set s = concat(s, '$10,000 to $20,000: ');
57
58   while row_not_found = false do
59     fetch invoices_cursor
60     into vendor_name_var, invoice_number_var, balance_due_var;
61
62     if balance_due_var >= 10000 and balance_due_var < 20000 then
63       set s = concat(s, balance_due_var, '|',
64         invoice_number_var, '|',
65         vendor_name_var, '///');
66     end if;
67   end while;
68 end;
69
70 close invoices_cursor;
71
72 -- display the string variable
73 select s as message;
74 end//
75
76 -- change statement delimiter from semicolon to double front slash

```

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The script editor displays the following code:

```
98  Lend//
99
100  -- change statement delimiter from semicolon to double front slash
101  delimiter ;
102
103  call test();
104
```

Below the editor is the 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The grid contains one row with the following data:

message
\$20,000 or more: \$10,000 to \$20,000: 19351.1...

On the right side, there is a 'Result Grid' button and a 'Read Only' indicator.