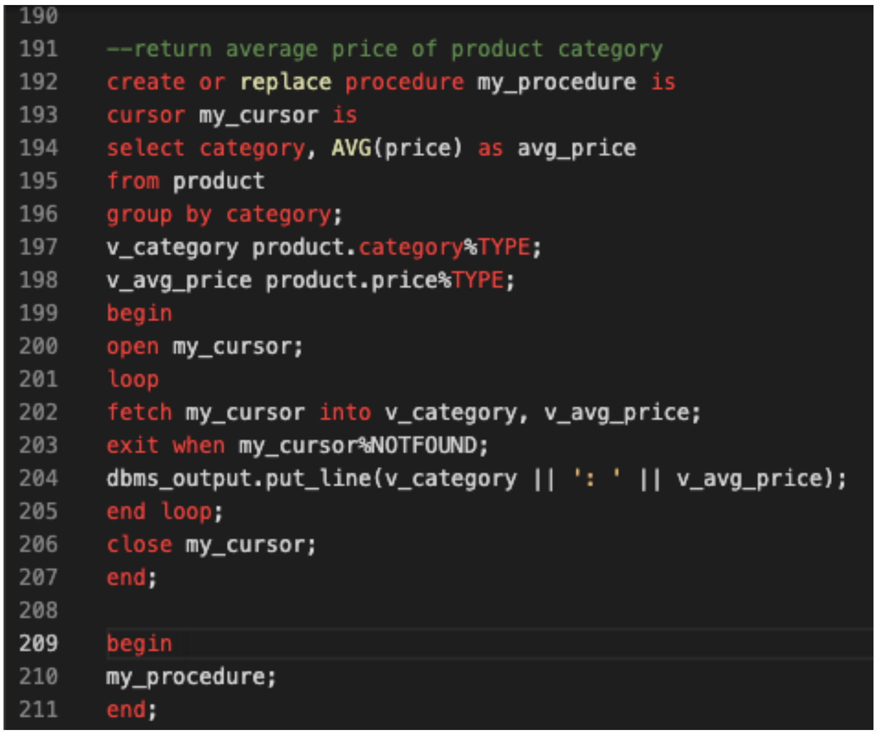
Procedures

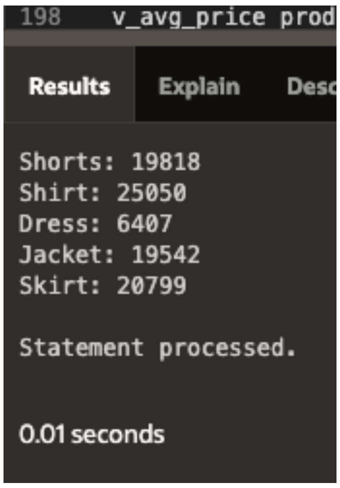
1) The first procedure returns us the average price of a product in a particular category. For example, we have a category of skirts, t-shirts, and so on. This procedure returns the average price of each category.

This is the code:



```
--return average price of product category
create or replace procedure my_procedure is
cursor my_cursor is
select category, AVG(price) as avg_price
from product
group by category;
v_category product.category%TYPE;
v_avg_price product.price%TYPE;
begin
open my_cursor;
loop
fetch my_cursor into v_category, v_avg_price;
exit when my_cursor%NOTFOUND;
dbms_output.put_line(v_category || ': ' || v_avg_price);
end loop;
close my_cursor;
end;

begin
my_procedure;
end;
```

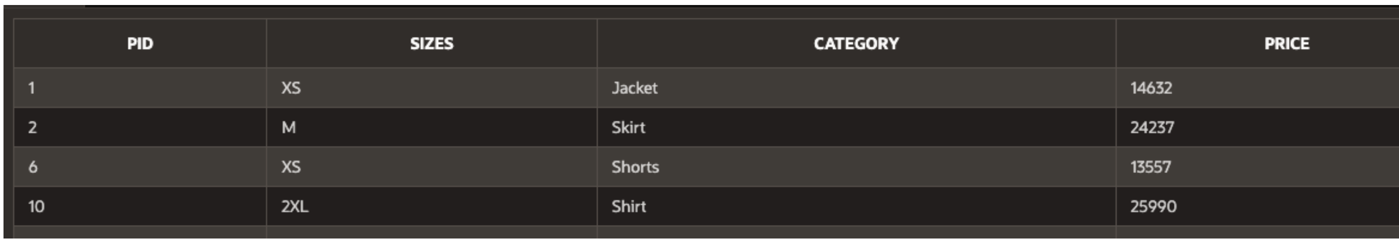
And this is the result of our function:



```
Shorts: 19818
Shirt: 25050
Dress: 6407
Jacket: 19542
Skirt: 20799

Statement processed.

0.01 seconds
```

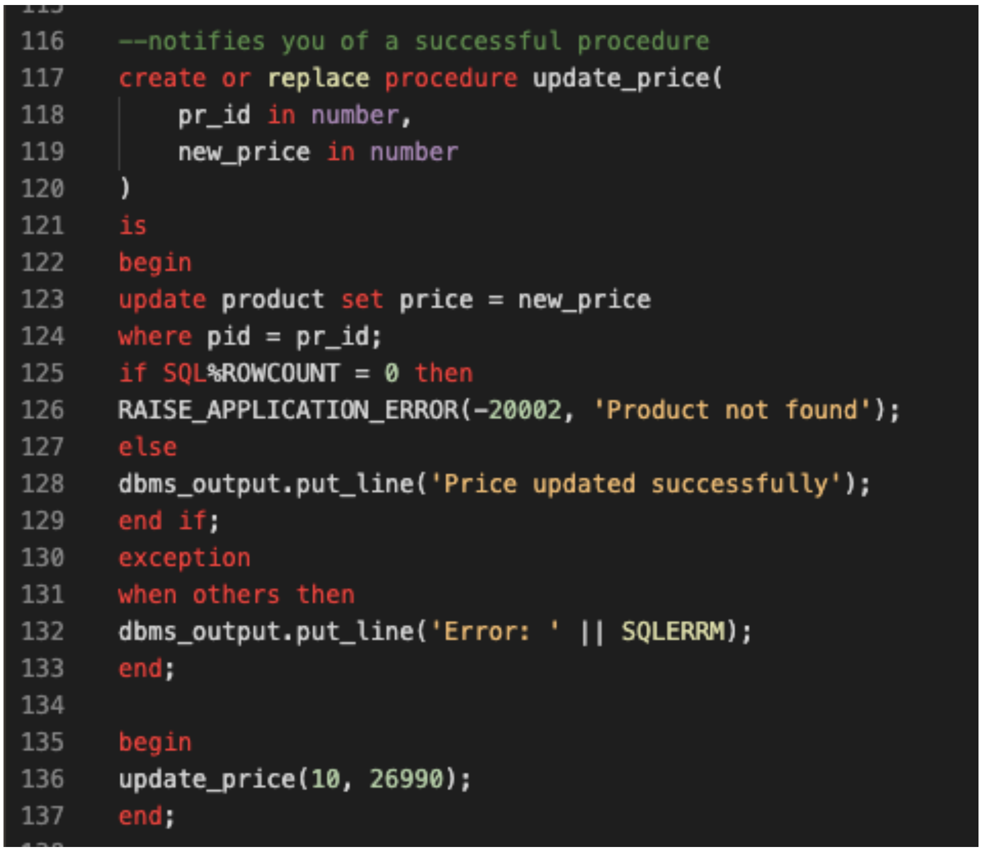
2) This procedure changes the price of a product and throws an error if the number of rows changed is 0 (through %ROWCOUNT).

Table look like this before update:



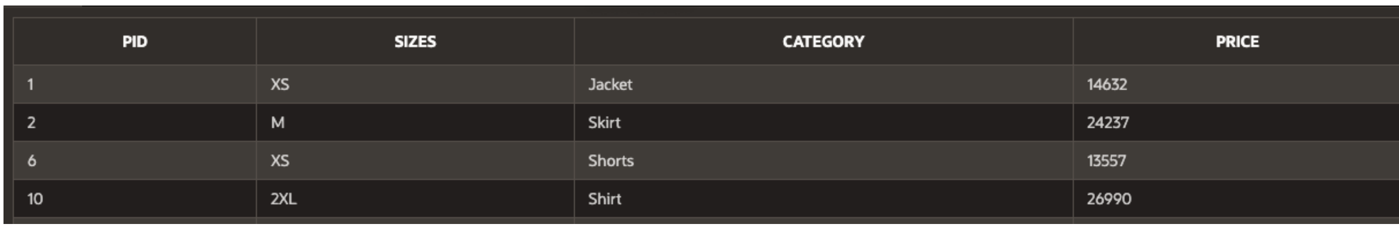
PID	SIZES	CATEGORY	PRICE
1	XS	Jacket	14632
2	M	Skirt	24237
6	XS	Shorts	13557
10	2XL	Shirt	25990

This is the code:



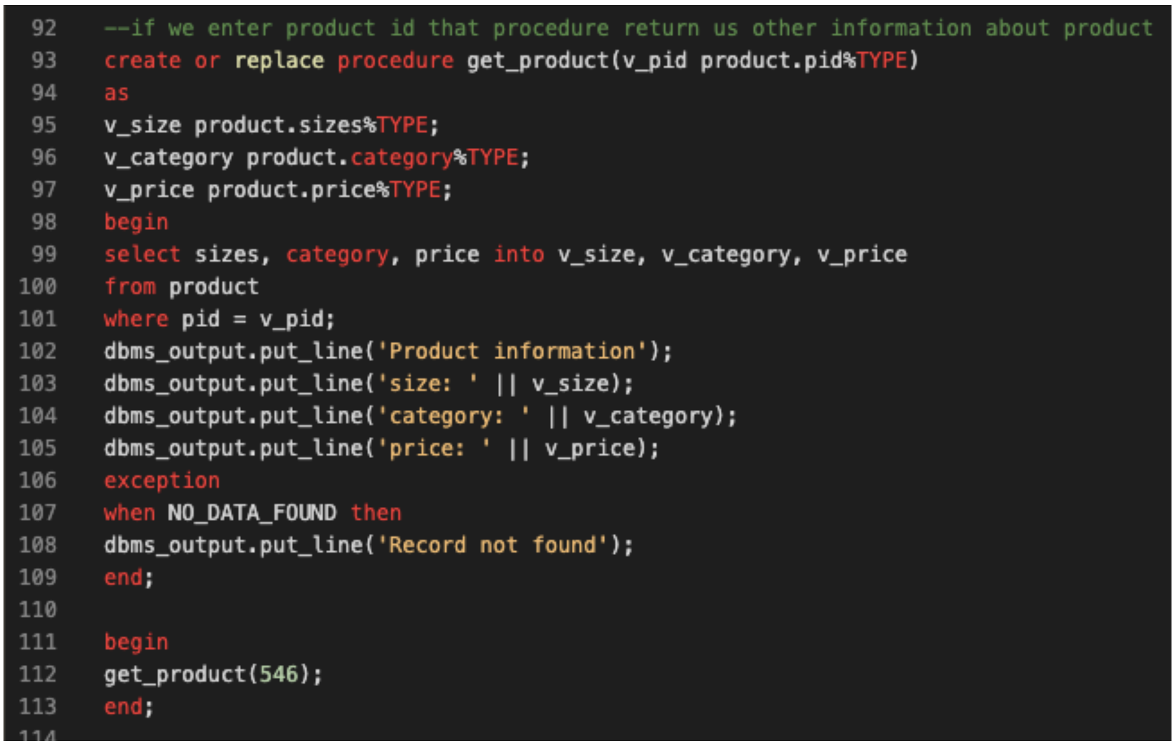
```
--notifies you of a successful procedure
create or replace procedure update_price(
    pr_id in number,
    new_price in number
)
is
begin
update product set price = new_price
where pid = pr_id;
if SQL%ROWCOUNT = 0 then
RAISE_APPLICATION_ERROR(-20002, 'Product not found');
else
dbms_output.put_line('Price updated successfully');
end if;
exception
when others then
dbms_output.put_line('Error: ' || SQLERRM);
end;

begin
update_price(10, 26990);
end;
```

And this is result of our function (You can see that price of product with PID 10 is 26990):



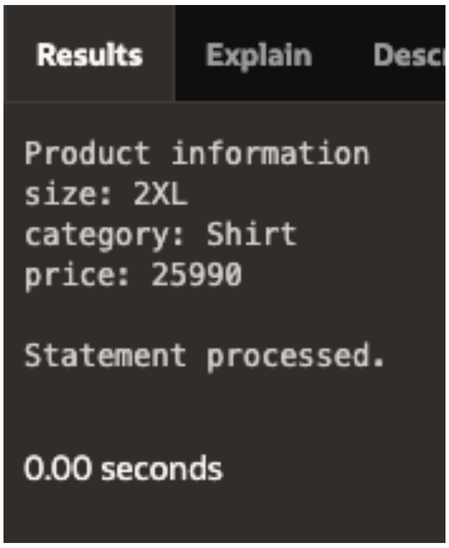
PID	SIZES	CATEGORY	PRICE
1	XS	Jacket	14632
2	M	Skirt	24237
6	XS	Shorts	13557
10	2XL	Shirt	26990

3) The following procedure returns all information about the product to us if we enter its ID:



```
--if we enter product id that procedure return us other information about product
create or replace procedure get_product(v_pid product.pid%TYPE)
as
v_size product.sizes%TYPE;
v_category product.category%TYPE;
v_price product.price%TYPE;
begin
select sizes, category, price into v_size, v_category, v_price
from product
where pid = v_pid;
dbms_output.put_line('Product information');
dbms_output.put_line('size: ' || v_size);
dbms_output.put_line('category: ' || v_category);
dbms_output.put_line('price: ' || v_price);
exception
when NO_DATA_FOUND then
dbms_output.put_line('Record not found');
end;

begin
get_product(546);
end;
```

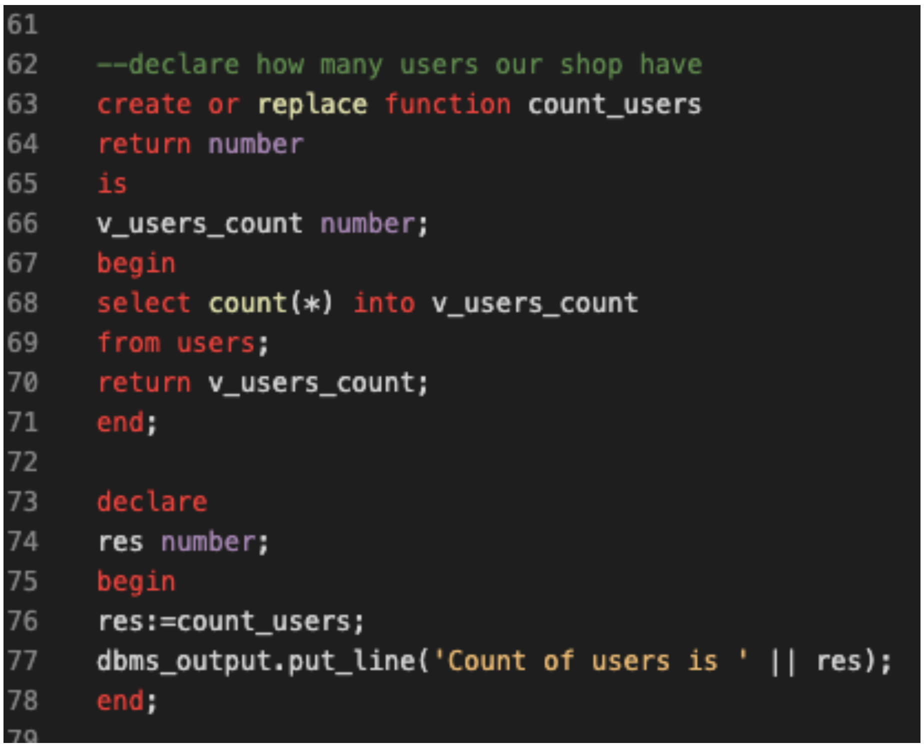
This is the result of procedure:



```
Product information
size: 2XL
category: Shirt
price: 25990

Statement processed.

0.00 seconds
```

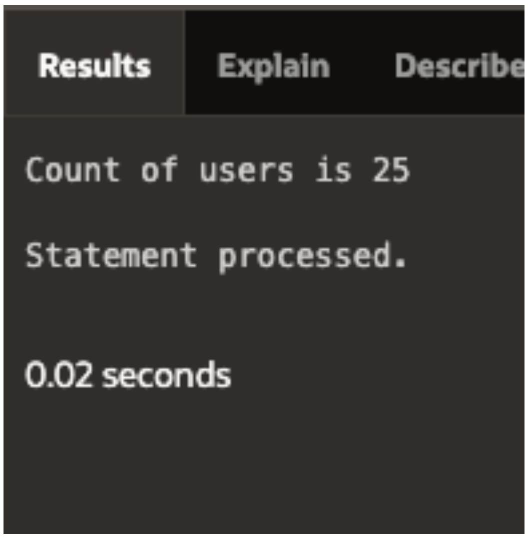
Functions

1) This function declare how many users our shop had(Count(*)):



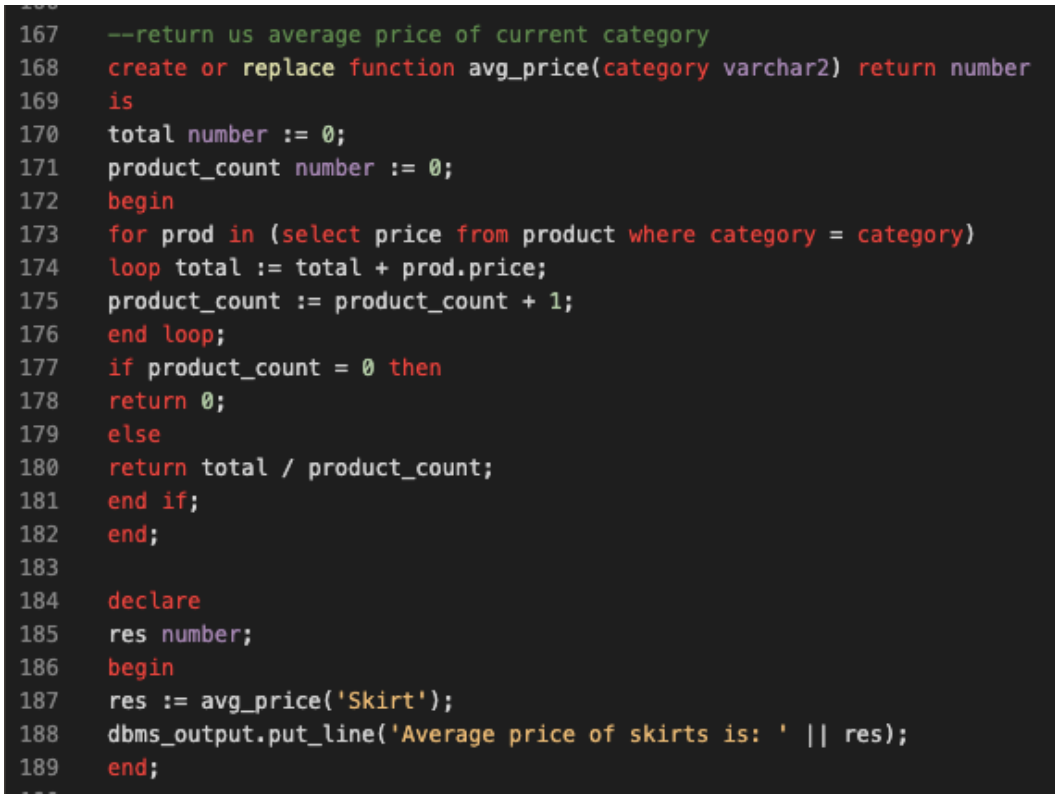
```
--declare how many users our shop have
create or replace function count_users
return number
is
v_users_count number;
begin
select count(*) into v_users_count
from users;
return v_users_count;
end;

declare
res number;
begin
res:=count_users;
dbms_output.put_line('Count of users is ' || res);
end;
```

This is the result of function:



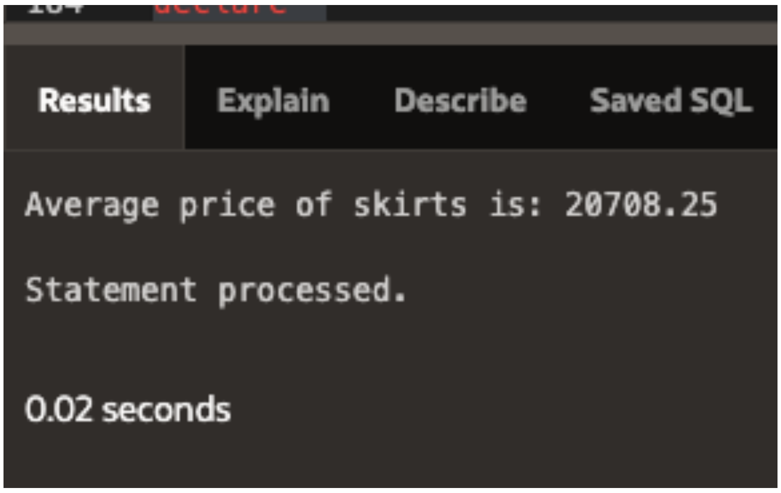
```
Count of users is 25

Statement processed.

0.02 seconds
```

2) This function returns average price of current category. For example, we have skirts category. This function returns us average price of skirts:



```
--return us average price of current category
create or replace function avg_price(category varchar2) return number
is
total number := 0;
product_count number := 0;
begin
for prod in (select price from product where category = category)
loop total := total + prod.price;
product_count := product_count + 1;
end loop;
if product_count = 0 then
return 0;
else
return total / product_count;
end if;
end;

declare
res number;
begin
res := avg_price('Skirt');
dbms_output.put_line('Average price of skirts is: ' || res);
end;
```

Result of this function is:



```
Average price of skirts is: 20708.25

Statement processed.

0.02 seconds
```