DBMS – 2

Temirlan Arystan

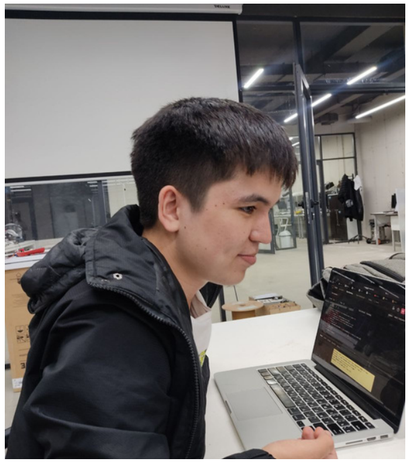
Student

Nurkhat Muratkan

Student

Dias Gaziz

Student



Sabyrzhan Rustembekov

Student

Our Team

Dias Gaziz

Sabyrzhan Rustembekov

Nurkhat Muratkhan

Temirlan Arystan

Online Book Store

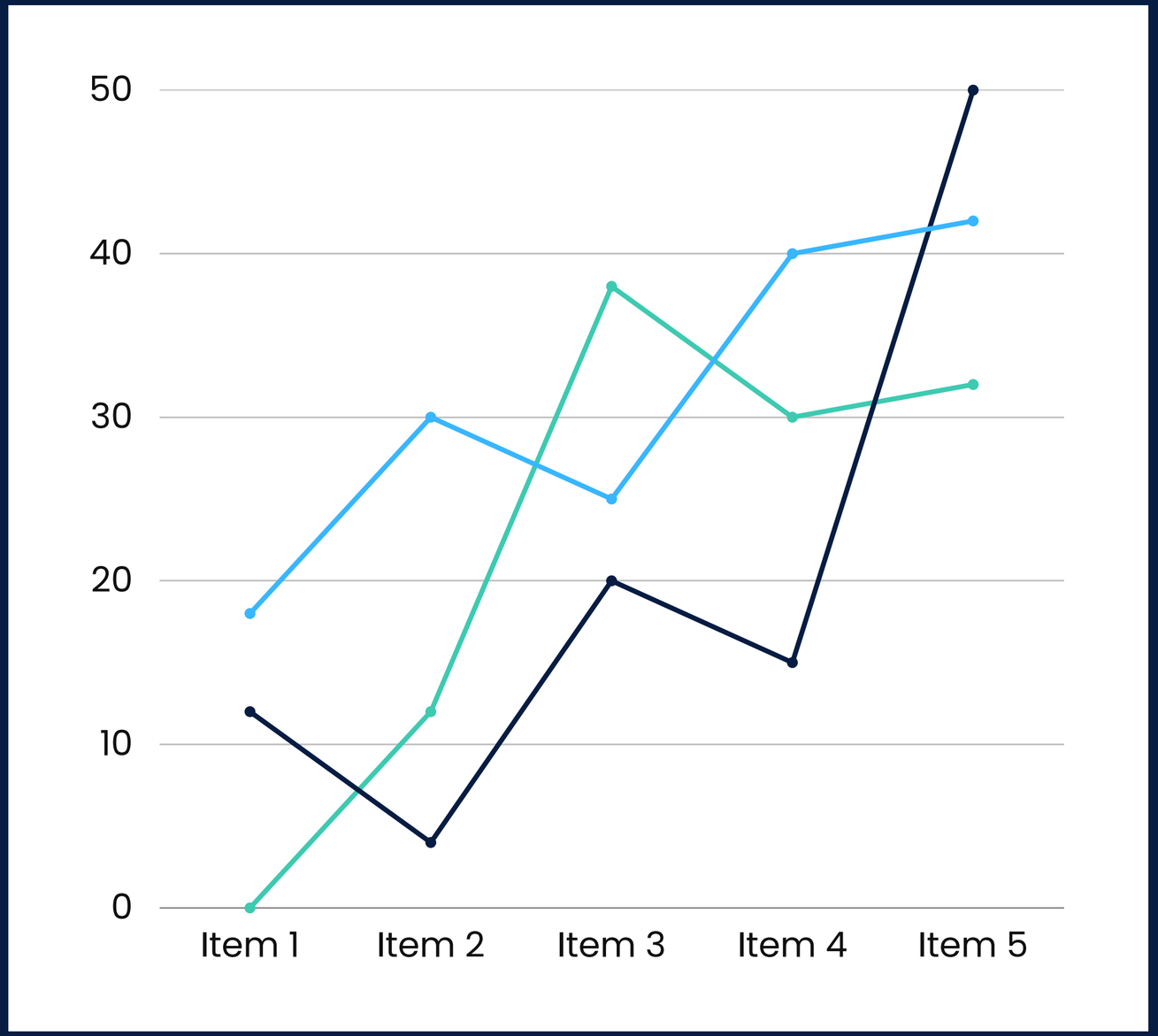
Description

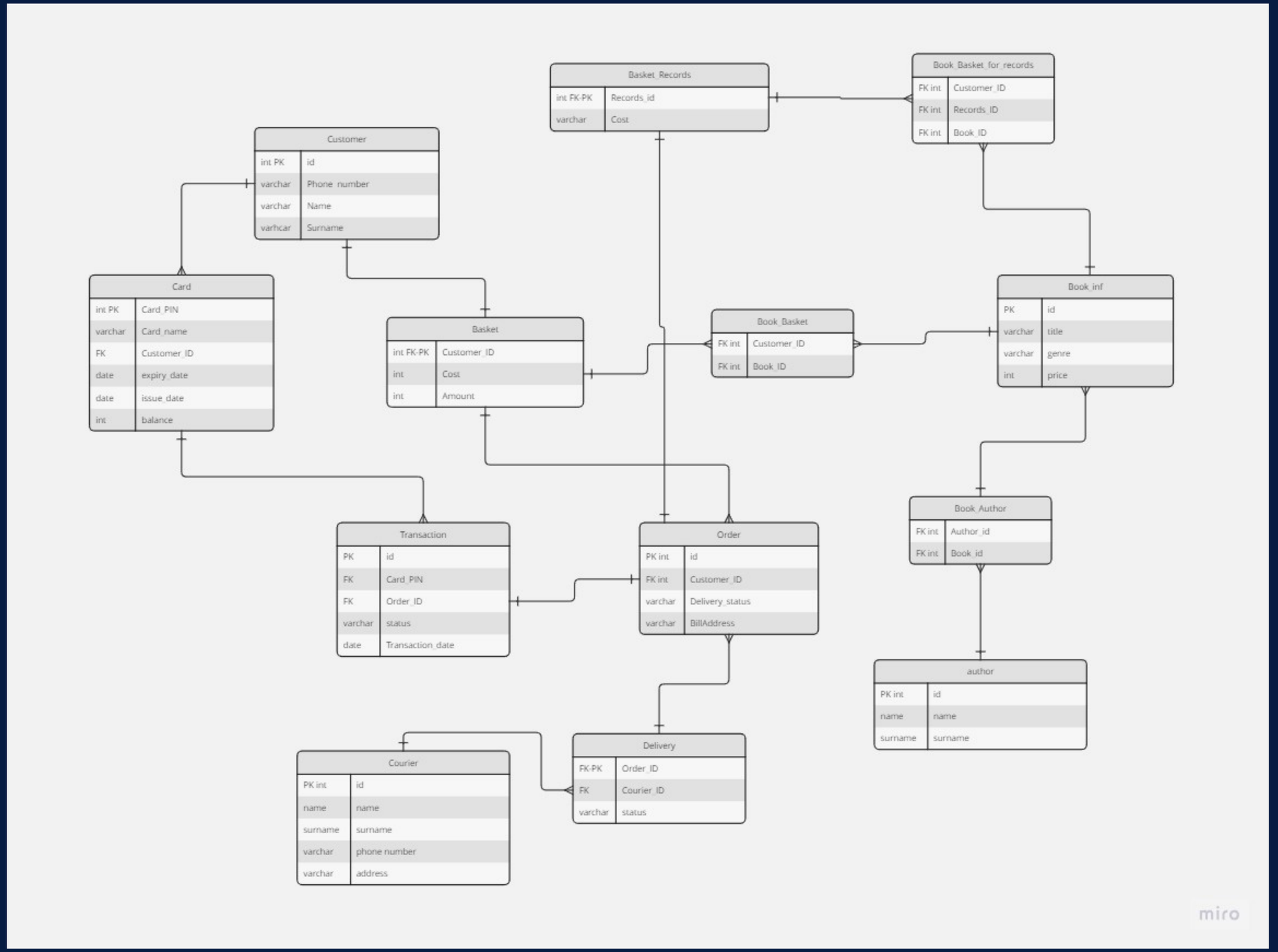
Our database allows the online bookstore to improve its activities. Facilitates management. The database tracks all events.

To do this, we have tables: Book_info, Customers, Basket, Book_Basket, Order, Delivery, Courier, Author, Book_Author, Card, Payment(Transactions), Basket_Records, Book_Basket_for_records.

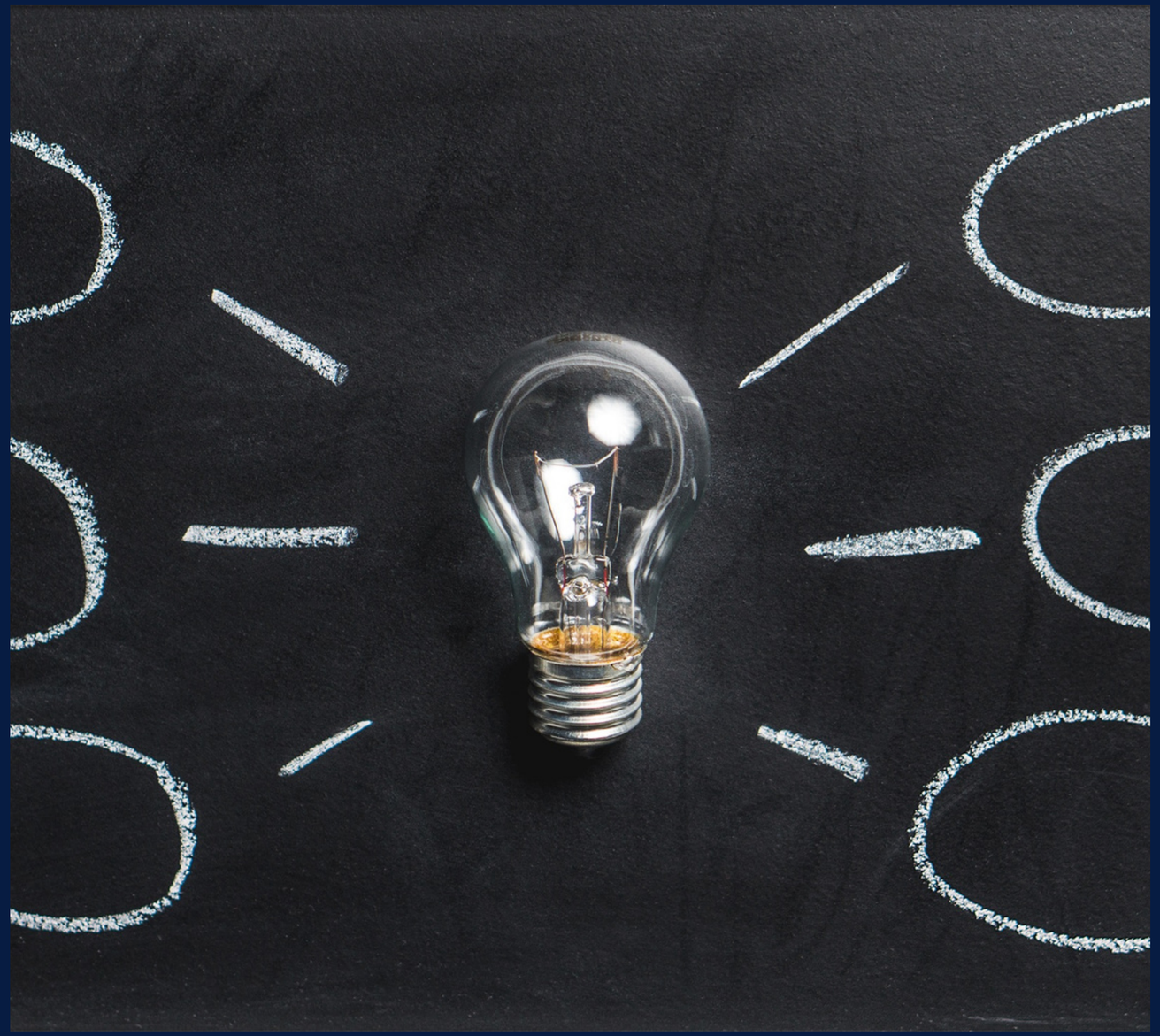
These tables perform the functions of courier delivery pickup or courier, income, sort by book authors, and much more. The end user of the database is the customer who buys the book. The problem of data obsolescence will be solved with the help of triggers. We came up with the idea of the project, and the Internet became the main source of resources.

ER-diagram




Basket_Records
int FK-PK
Records_id
varchar
Cost
Book_Basket_for_records
FK int
Customer_ID
FK int
Records_ID
FK int
Book_ID
Customer
int PK
id
varchar
Phone_number
varchar
Name
varhcar
Surname
Card
int PK
Card_PIN
varchar
Card_name
FK
Customer_ID
date
expiry_date
date
issue_date
int
balance
Basket
int FK-PK
Customer_ID
int
Cost
int
Amount
Book_Basket
FK int
Customer_ID
FK int
Book_ID
Book_inf
PK
id
varchar
title
varchar
genre
int
price
Book_Author
FK int
Author_id
FK int
Book_id
Transaction
PK
id
FK
Card_PIN
FK
Order_ID
varchar
status
date
Transaction_date
Order
PK int
id
FK int
Customer_ID
varchar
Delivery_status
varchar
BillAddress
author
PK int
id
name
name
surname
surname
Courier
PK int
id
name
name
surname
surname
varchar
phone number
varchar
address
Delivery
FK-PK
Order_ID
FK
Courier_ID
varchar
status

Procedures

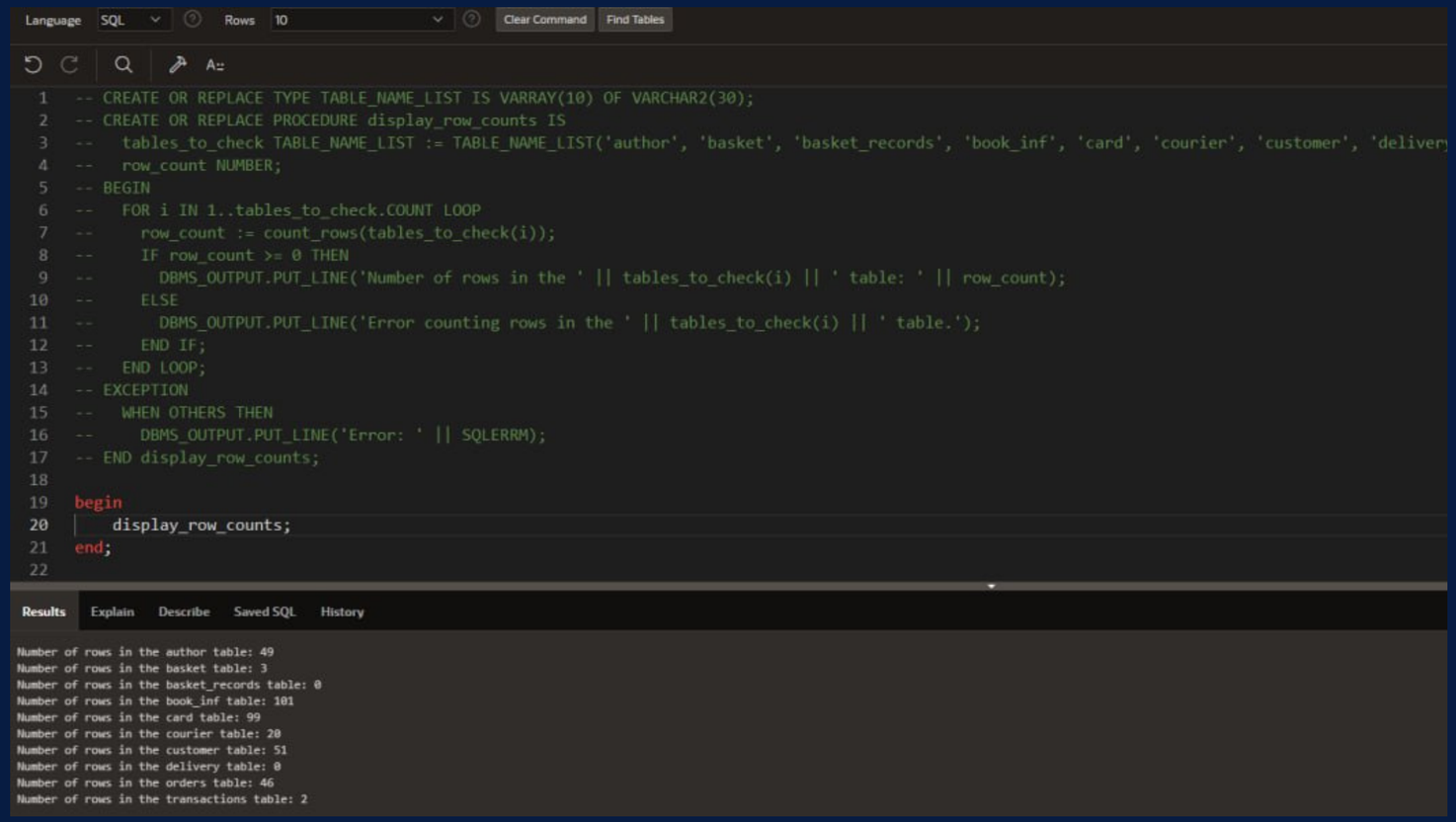


This procedure removes row from Transaction, Basket, Book_basket tables if the transaction status is set to "In expectation" and the time of adding to the basket has exceeded 3 days.

```
create or replace PROCEDURE delete_unpaid_orderss AS
CURSOR c_order IS
SELECT tr.order_id, tr.status, od.delivery_status, od.customer_id
FROM Transactions tr
INNER JOIN Delivery dl
ON dl.order_id = tr.order_id
INNER JOIN Orders od
ON od.id = dl.order_id
WHERE tr.transaction_date < (SYSDATE - 3)
AND tr.status = 'In expectation';
BEGIN
FOR order_rec IN c_order LOOP
DELETE FROM Book_basket WHERE customer_id = order_rec.customer_id;
DELETE FROM Basket WHERE customer_id = order_rec.customer_id;
IF order_rec.delivery_status = 'courier' then
DELETE FROM Delivery WHERE order_id = order_rec.order_id;
END IF;
DELETE FROM Transactions WHERE order_id = order_rec.order_id;
DBMS_OUTPUT.PUT_LINE('Order deleted is id: ' || order_rec.order_id);
END LOOP;
END;
```

Displays the row sum of all tables

```
create or replace PROCEDURE display_row_counts IS
tables_to_check TABLE_NAME_LIST :=
TABLE_NAME_LIST('author', 'basket', 'basket_records',
'book_inf', 'card', 'courier', 'customer', 'delivery', 'orders',
'transactions');
row_count NUMBER;
BEGIN
FOR i IN 1..tables_to_check.COUNT LOOP
row_count := count_rows(tables_to_check(i));
IF row_count >= 0 THEN
DBMS_OUTPUT.PUT_LINE('Number of rows in the ' ||
tables_to_check(i) || ' table: ' || row_count);
ELSE
DBMS_OUTPUT.PUT_LINE('Error counting rows in the ' ||
tables_to_check(i) || ' table.');
END IF;
END LOOP;
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END display_row_counts;
```



The procedure is done on sql%rowcount. This procedure does update by specifying customer_id, we do update amount book

```
create or replace PROCEDURE update_book_amount (
p_customer_id IN NUMBER,
p_new_amount IN NUMBER)
IS BEGIN
UPDATE basket
SET amount = p_new_amount
WHERE customer_id = p_customer_id;

IF SQL%ROWCOUNT > 0 THEN
DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' rows updated.');
ELSE
DBMS_OUTPUT.PUT_LINE('No rows updated.');
END IF;
END;

For example:
BEGIN
update_book_amount(4, 7);
END;
```

Packages

This package totals a row of the table

Specification:

```
create or replace PACKAGE total_profit AS
PROCEDURE print_total;
END total_profit;
```

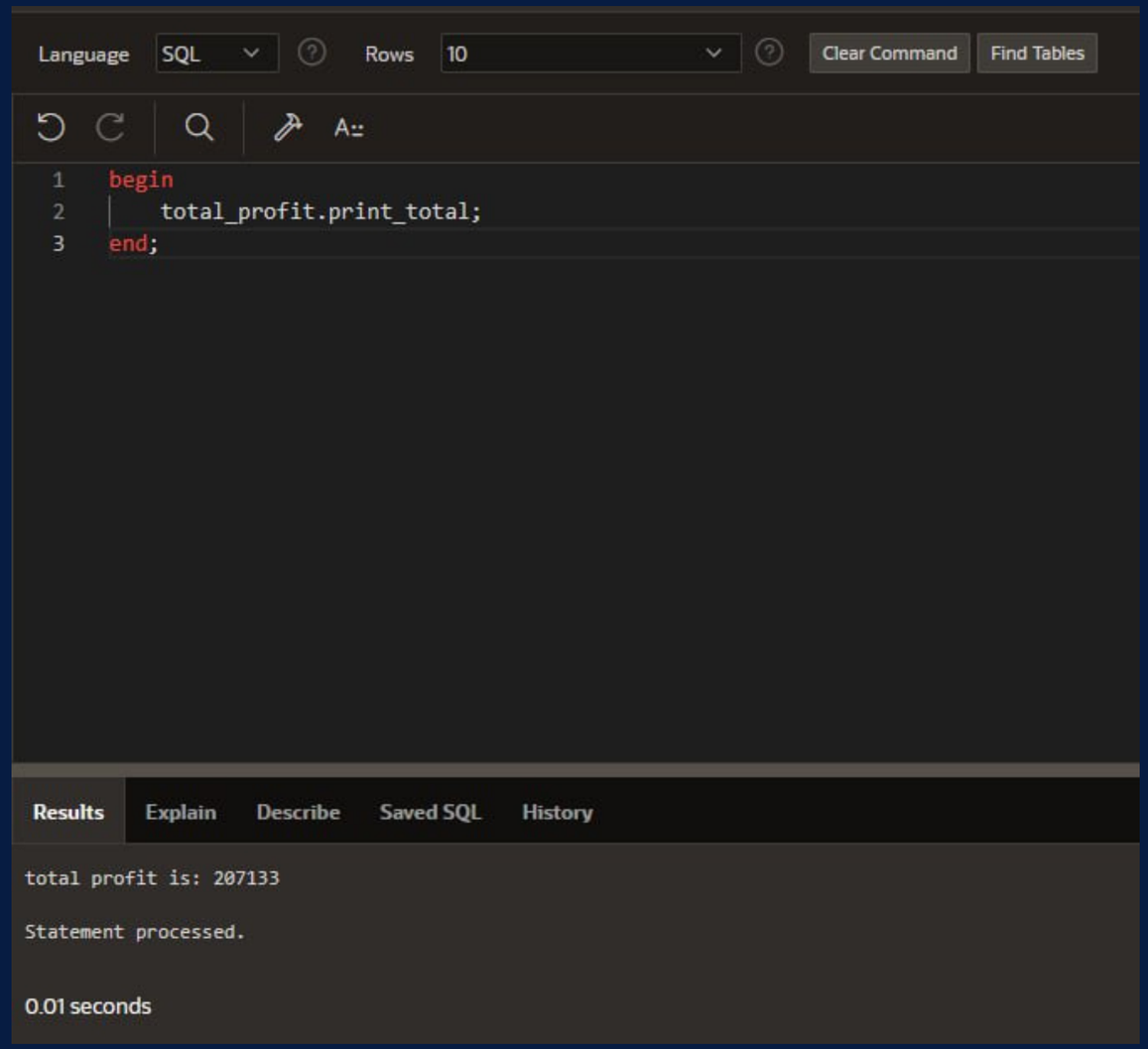
BODY:

```
create or replace PACKAGE BODY total_profit AS

PROCEDURE print_total IS
 sum_price NUMBER(10);
BEGIN
SELECT SUM(cost) INTO sum_price FROM basket_records;
dbms_output.put_line('total profit is: ' || TO_CHAR(sum_price));
END print_total;

END total_profit;
```

```
begin
total_profit.print_total;
end;
```



This function combines two functions, the first (to get a list of all books by a certain author), the second (to get a list of all orders for a certain client) and three procedures add_book, update_bool, delete_book

BOOKMANAGMENT

Specification | Body | Dependencies | Errors | Grants

Download | Save and Compile | Drop | Refresh

```
1  create or replace PACKAGE BODY BookManagment AS
2
3    FUNCTION get_books_by_author(p_author_id NUMBER) RETURN SYS_REFCURSOR IS
4      book_cursor SYS_REFCURSOR;
5    BEGIN
6      OPEN book_cursor FOR
7        SELECT b.*
8        FROM BOOK_INF b
9        JOIN BOOK_AUTHOR ba ON b.ID = ba.BOOK_ID
10       WHERE ba.AUTHOR_ID = p_author_id;
11     RETURN book_cursor;
12   END get_books_by_author;
13
14   FUNCTION get_orders_by_customer(p_customer_id NUMBER) RETURN SYS_REFCURSOR IS
15     order_cursor SYS_REFCURSOR;
16   BEGIN
17     OPEN order_cursor FOR
18       SELECT o.*
19       FROM ORDERS o
20       WHERE o.CUSTOMER_ID = p_customer_id;
21     RETURN order_cursor;
22   END get_orders_by_customer;
23
24   PROCEDURE add_book(p_title VARCHAR2, p_genre VARCHAR2, p_price NUMBER, p_author_ids SYS.ODCINUMBERLIST) IS
25     new_book_id NUMBER;
26   BEGIN
27     INSERT INTO BOOK_INF (TITLE, GENRE, PRICE)
28     VALUES (p_title, p_genre, p_price)
29     RETURNING ID INTO new_book_id;
```

BOOKMANAGMENT

Specification | Body | Dependencies | Errors | Grants

Download | Save and Compile | Drop | Refresh

```
30
31     FOR i IN 1 .. p_author_ids.COUNT LOOP
32       INSERT INTO BOOK_AUTHOR (AUTHOR_ID, BOOK_ID)
33       VALUES (p_author_ids(i), new_book_id);
34     END LOOP;
35
36     COMMIT;
37   END add_book;
38
39   PROCEDURE update_book(p_book_id NUMBER, p_title VARCHAR2, p_genre VARCHAR2, p_price NUMBER) IS
40   BEGIN
41     UPDATE BOOK_INF
42     SET TITLE      = p_title,
43         GENRE      = p_genre,
44         PRICE      = p_price
45     WHERE ID = p_book_id;
46
47     COMMIT;
48   END update_book;
49
50   PROCEDURE delete_book(p_book_id NUMBER) IS
51   BEGIN
52     DELETE FROM BOOK_AUTHOR WHERE BOOK_ID = p_book_id;
53     DELETE FROM BOOK_INF WHERE ID = p_book_id;
54
55     COMMIT;
56   END delete_book;
57
58 END BookManagment;
```

FUNCTIONS

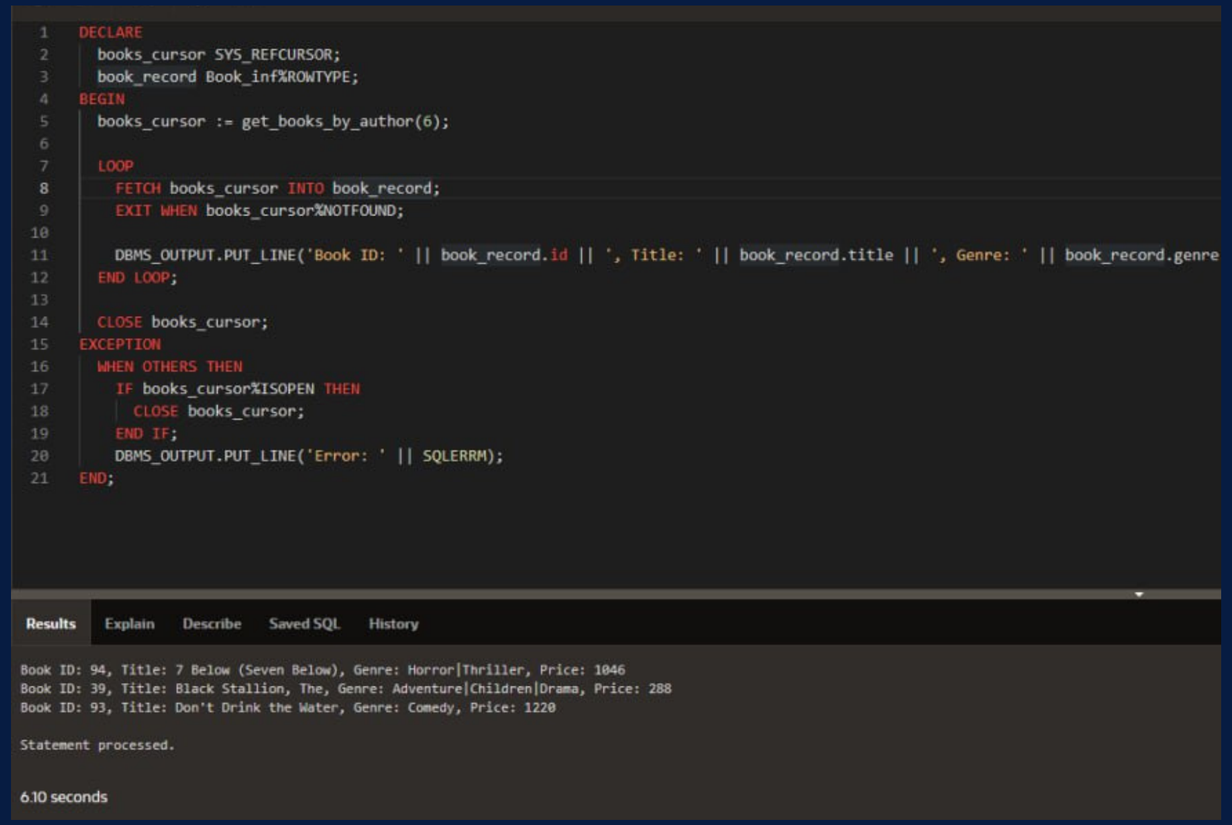
This function reads every row of the table

```
create or replace FUNCTION count_rows(p_table_name IN VARCHAR2) RETURN NUMBER IS
row_count NUMBER;
sql_query VARCHAR2(1000);
BEGIN
sql_query := 'SELECT COUNT(*) FROM ' || p_table_name;
EXECUTE IMMEDIATE sql_query INTO row_count;
RETURN row_count;
EXCEPTION
WHEN OTHERS THEN
RETURN -1;
END count_rows;
```

This function will output all the information about the author which book he wrote if we give the author's ID function

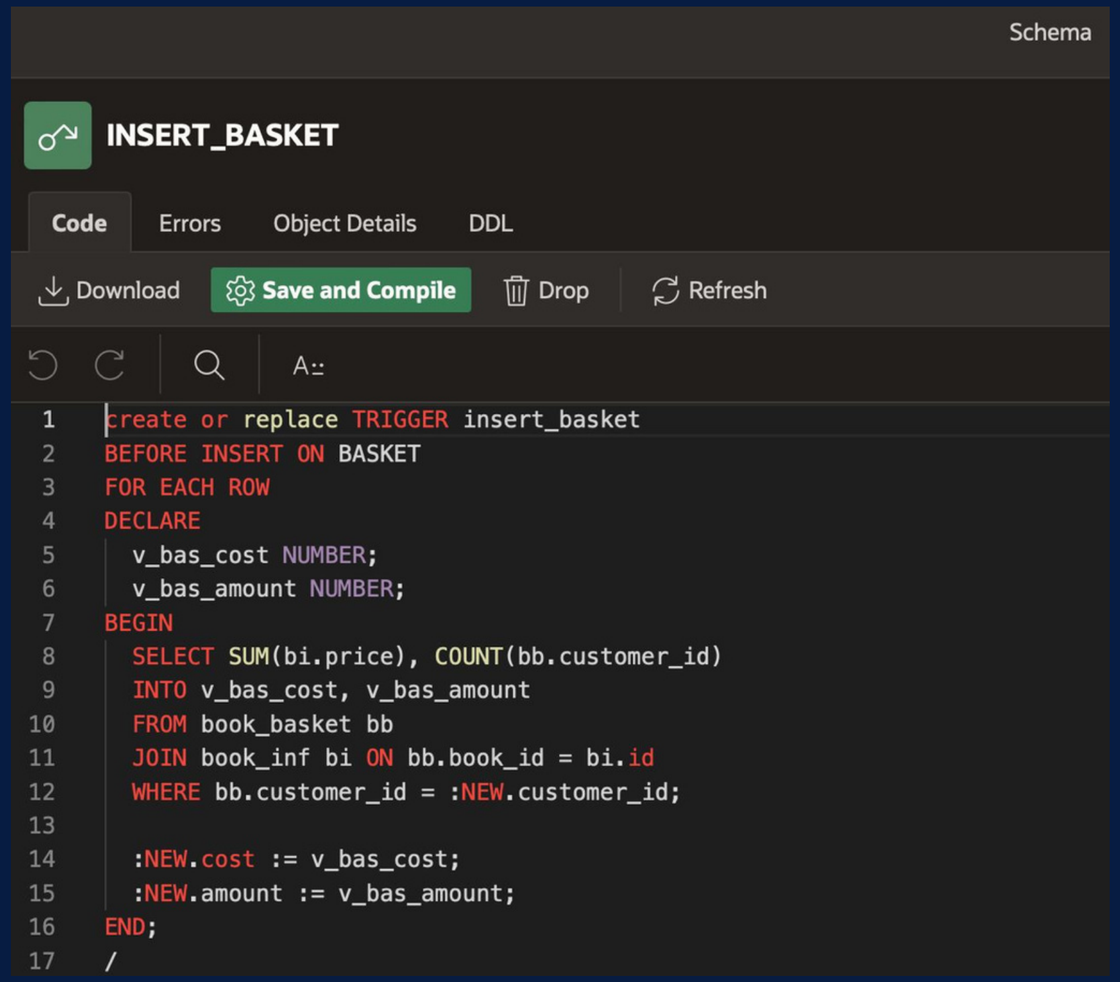
```
create or replace FUNCTION get_books_by_author(p_author_id NUMBER)
RETURN SYS_REFCURSOR
IS
books_cursor SYS_REFCURSOR;
BEGIN
OPEN books_cursor FOR
SELECT b.id, b.title, b.genre, b.price
FROM Book_inf b
JOIN book_author ba ON b.id = ba.book_id
WHERE ba.author_id = p_author_id;

RETURN books_cursor;
EXCEPTION
WHEN OTHERS THEN
RAISE;
END get_books_by_author;
```



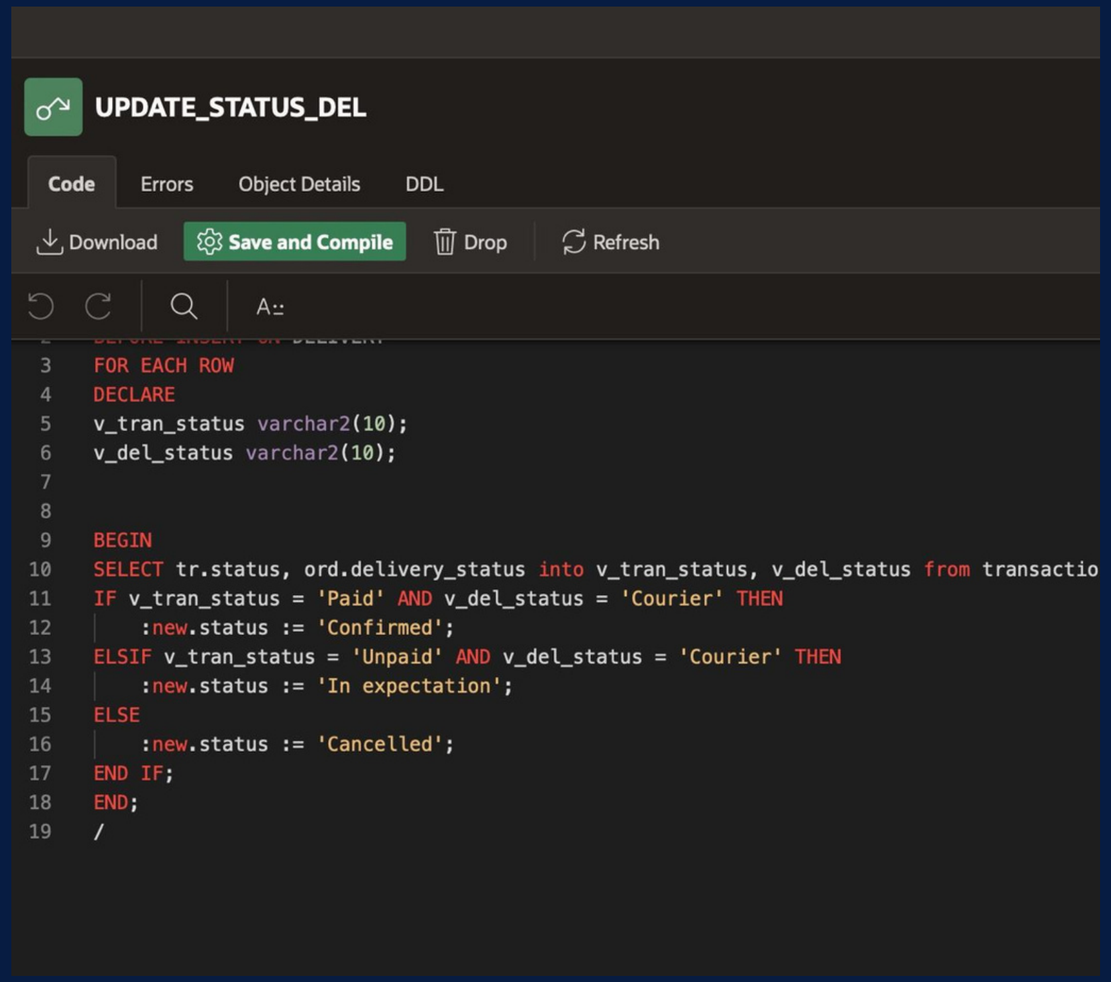
TRIGGERS

Trigger #1



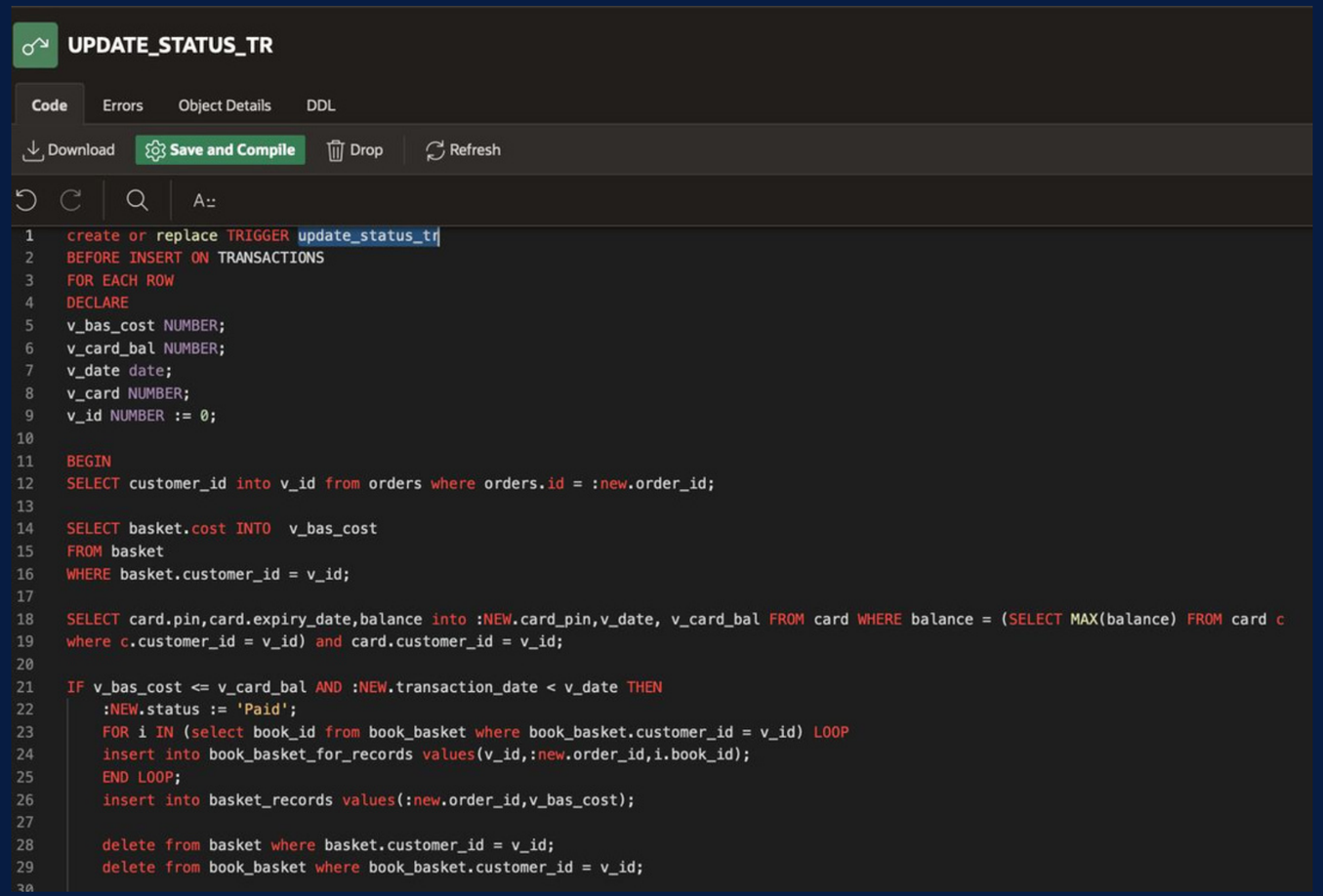
TRIGGER insert_basket
This trigger is needed for basket, just add the ID and get in cost recalculate all the amounts of the selected books and recalculate how many books the user took

Trigger #2



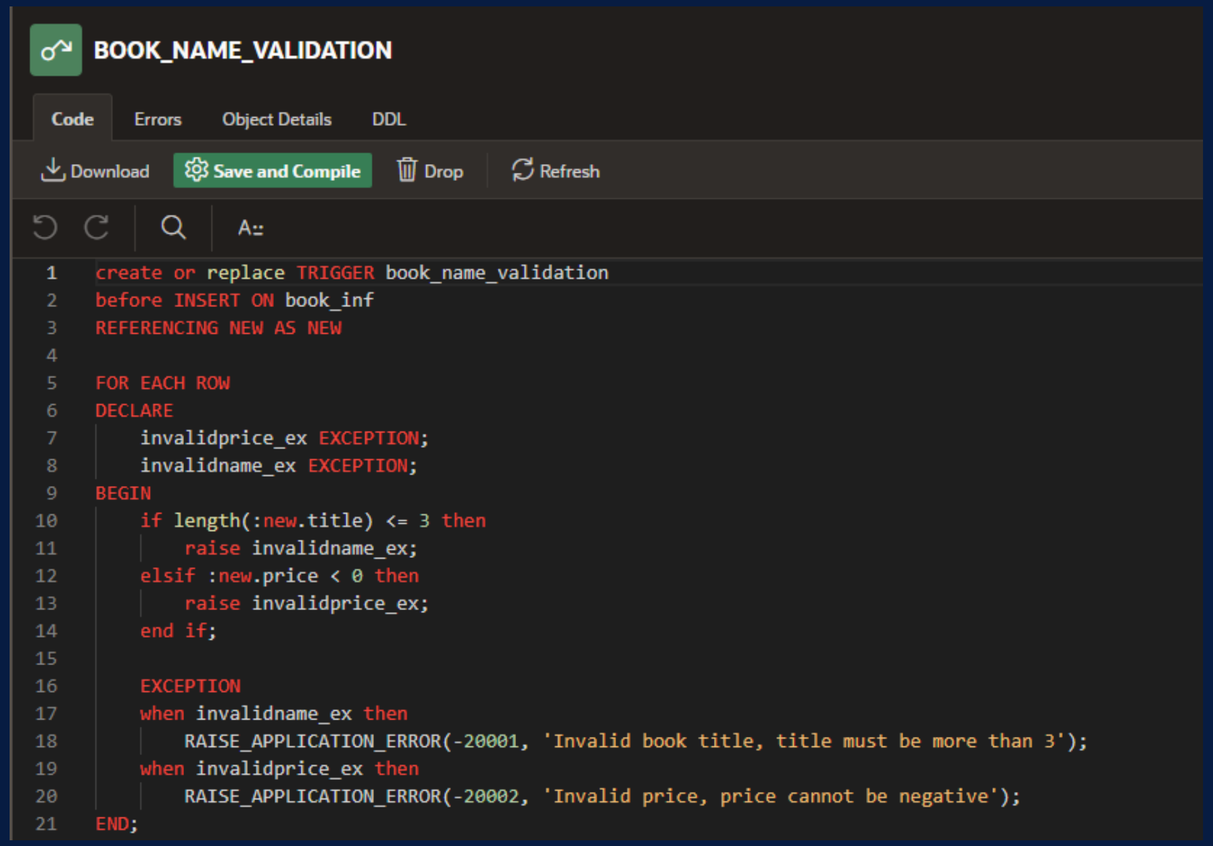
TRIGGER update_status_del Checks the delivery table of the courier changes its status that the customer took or is waiting for an order or canceled we check through the table transaction where his status is paid or not paid depends on the trigger and should not be self-expert should be to the address

Trigger #3



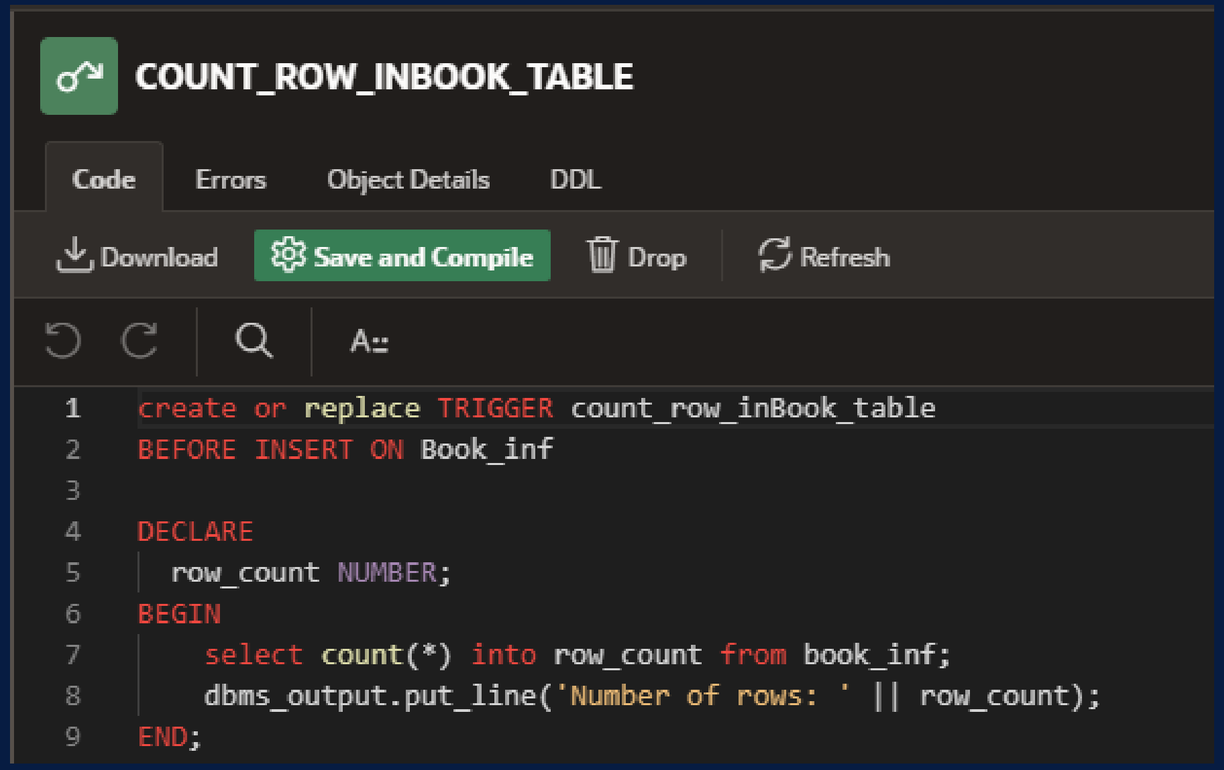
TRIGGER update_status_tracking
This trigger is needed if the user is to find out what he paid or not if he paid then we remove it from the basket and add it to the records we will check through what he has enough money in the card to buy a book and a card the expiration date has not yet passed

Trigger #4



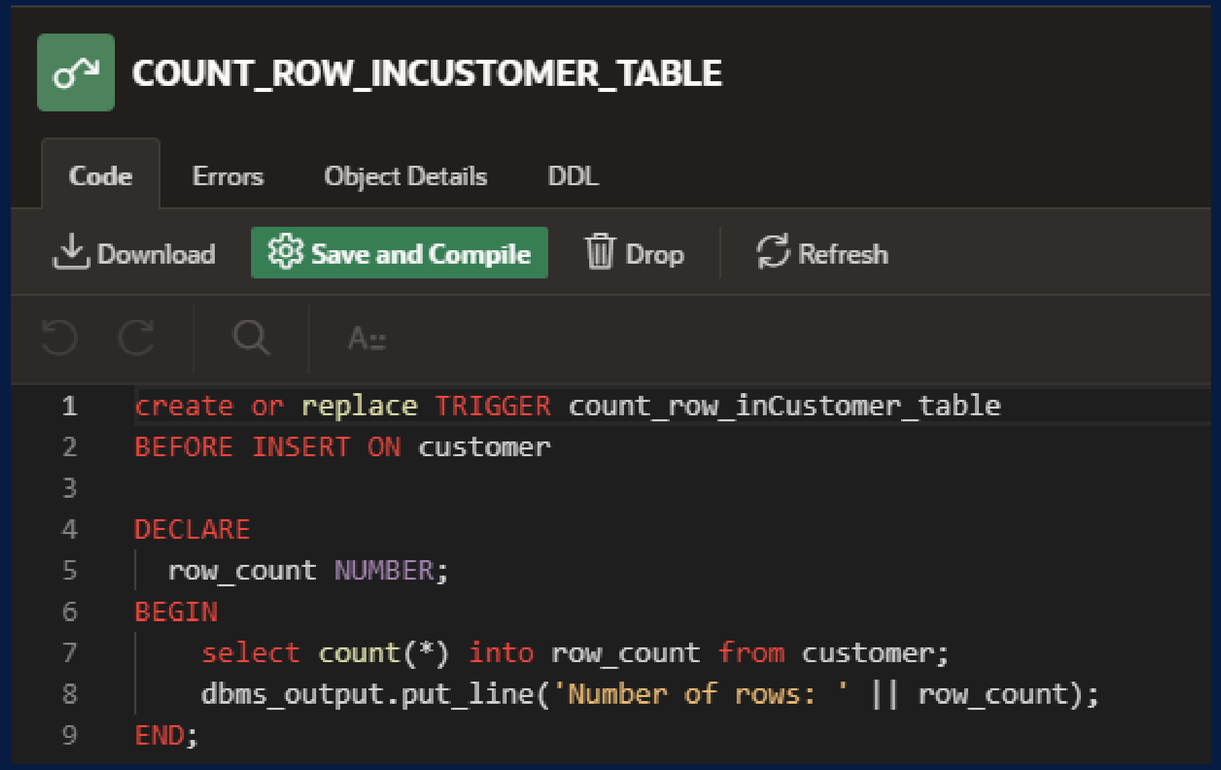
This trigger check book name and price for validation. If length of book name <= 3 or price is negative we catch user-defined exception.

Trigger #5

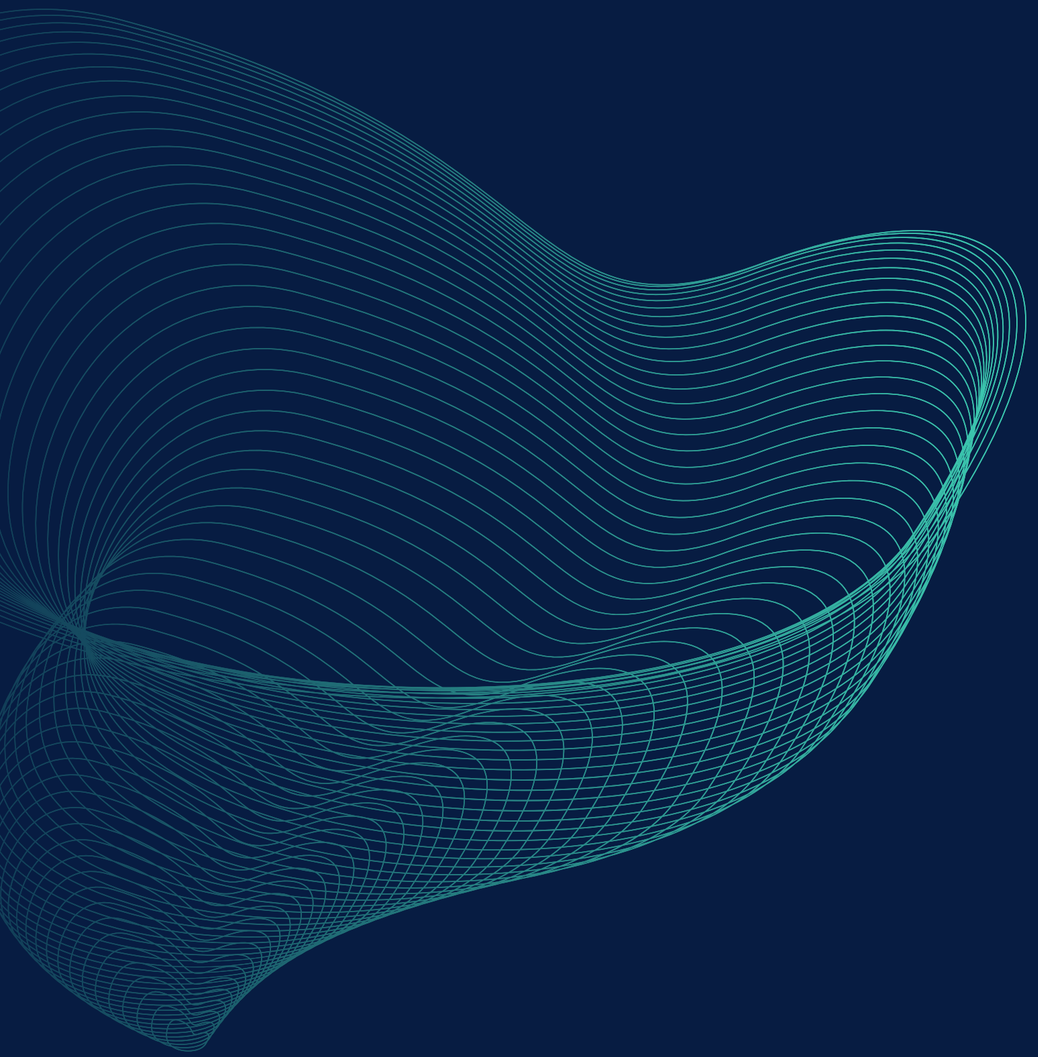


This triggers counts row numbers before inserting new row.

Trigger #6



This triggers counts row numbers before inserting new row.



Thank you for attention!