This presentation has live captioning.

DBMS Project

Products Shop

DBMS project for e-commerce management
As part of our university curriculum, we did this project for database management systems2 (DBMS2) - INF 305.
This project contains a theoretical part, as well as an implementation in PL/SQL.
If you like it, say "Yeeeeeeeeeeeeeeeeee" ⭐ this

Name of Project:	Presented By:	Presented To:
Online Shop	Mukhangali Ernar, Jumabay Erlan, Zhumadilla Nurdaulet	For teacher

Content

Content

Overview

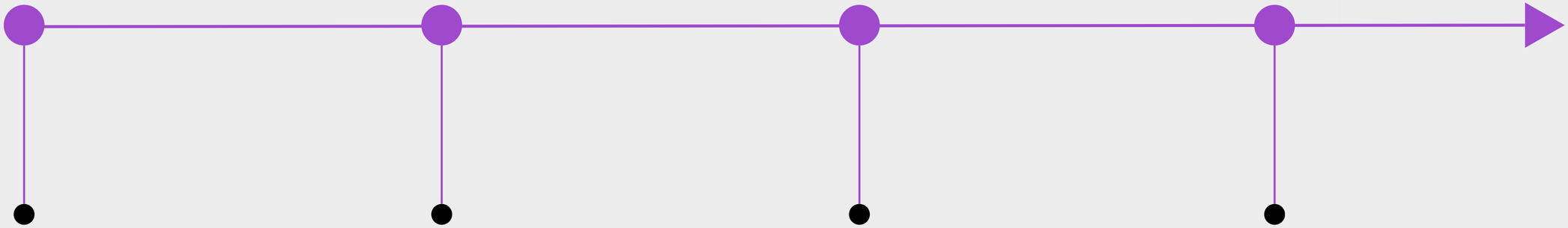
Overview

With the rise of e-commerce, it's becoming increasingly vital for sellers to establish an online presence. In fact, transitioning from a traditional brick-and-mortar store to an online sales model has become a necessity for many small clothing sellers. As an engineer, one of our primary responsibilities is to facilitate this transition and make it as seamless as possible. This is where a robust and efficient database system comes into play. Our project aims to develop a comprehensive database that empowers small clothing sellers to sell their products online with ease and efficiency. With this database in place, sellers can enjoy a hassle-free online shopping experience, which can translate into increased sales and revenue.

Basic Structure

Functional requirement

Main function:



1st

The buyer enters the online site.

2nd

The customer can add or remove an item from the cart.

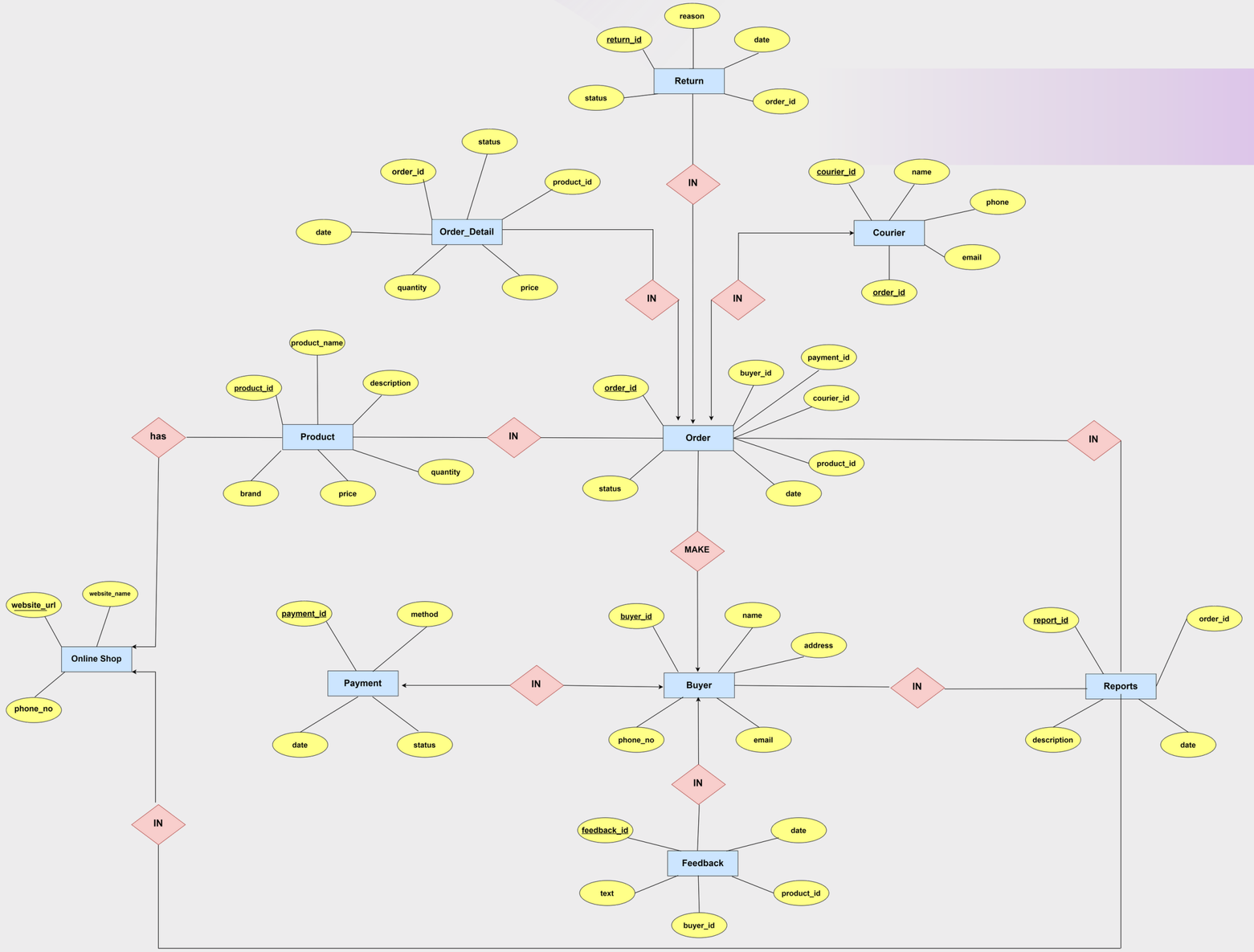
3rd

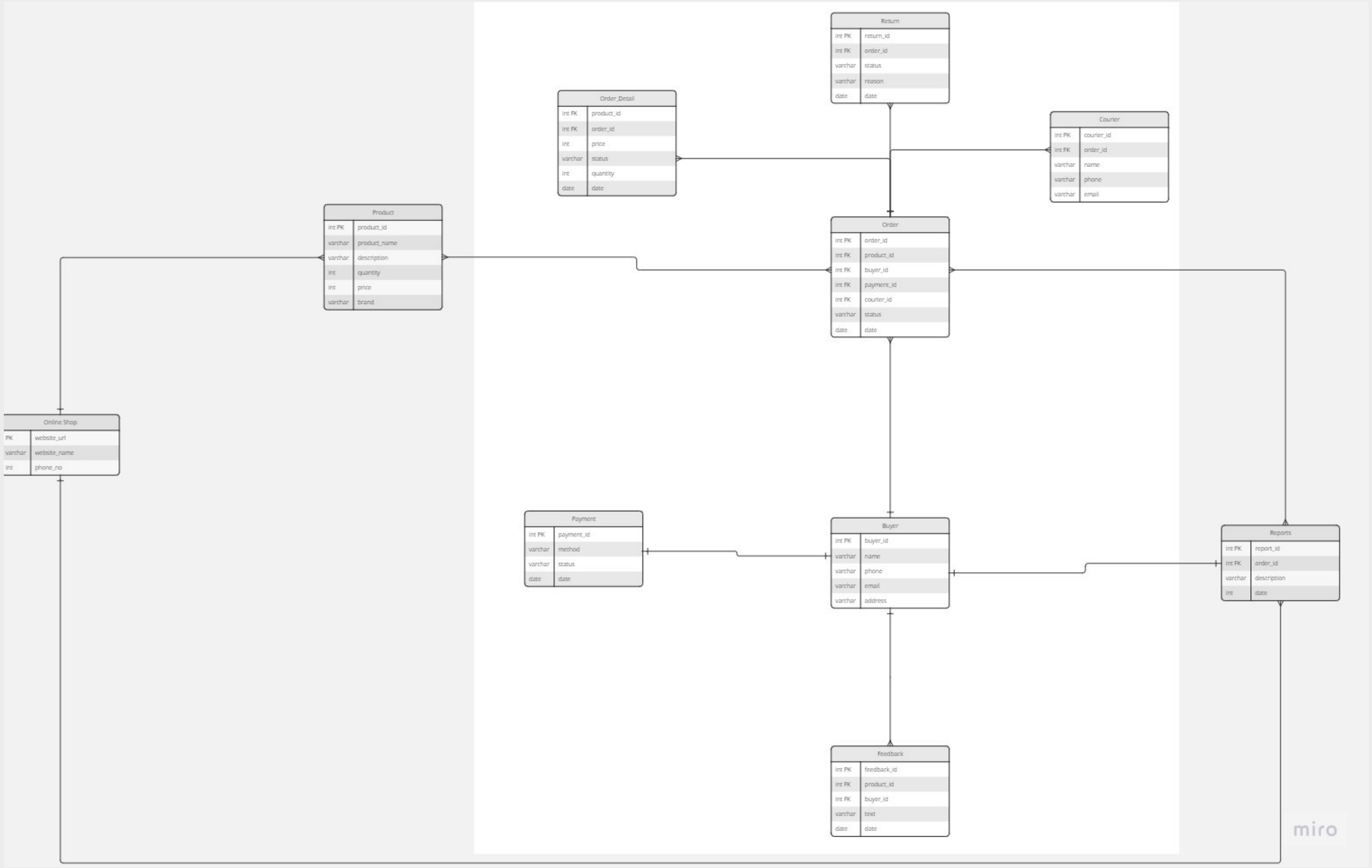
The buyer can view the total cost of the item in the cart, but not purchased.

4th

The administrator can view the total profit generated from the website.

Entity Relation Diagram




Return
int PK return_id
int FK order_id
varchar status
varchar reason
date date
Order_Detail
int FK product_id
int FK order_id
int price
varchar status
int quantity
date date
Courier
int PK courier_id
int FK order_id
varchar name
varchar phone
varchar email
Product
int PK product_id
varchar product_name
varchar description
int quantity
int price
varchar brand
Order
int PK order_id
int FK product_id
int FK buyer_id
int FK payment_id
int FK courier_id
varchar status
date date
Online Shop
PK website_url
varchar website_name
int phone_no
Payment
int PK payment_id
varchar method
varchar status
date date
Buyer
int PK buyer_id
varchar name
varchar phone
varchar email
varchar address
Reports
int PK report_id
int FK order_id
varchar description
int date
Feedback
int PK feedback_id
int FK product_id
int FK buyer_id
varchar text
date date
miro

FUNCTION

```sql
CREATE OR REPLACE FUNCTION get_total_orders_for_buyer(buyer IN NUMBER)
RETURN NUMBER
IS
  total_orders NUMBER;
BEGIN
  SELECT COUNT(*) INTO total_orders FROM ORDERS WHERE BUYER_ID = buyer;
  RETURN total_orders;
  IF SQL%ROWCOUNT=0 then
    raise NO_DATA_FOUND;
    END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No orders found for buyer with ID ' || buyer);
    RETURN 0;
END;
--
declare
n number;
Begin
n := get_total_orders_for_buyer(4);
 DBMS_OUTPUT.PUT_LINE('Total orders in buyer:  '||n);
end;
```

Results | Explain | Describe | Saved SQL | History

```
Total orders in buyer:  2

Statement processed.
```

0.01 seconds

Each customer has his own order

PROCEDURE

```
1  CREATE OR REPLACE PROCEDURE cancel_order (order_id IN ORDERS.order_id%TYPE)
2  IS
3  BEGIN
4    DELETE FROM ORDERDETAIL WHERE order_id = cancel_order.order_id;
5
6    DELETE FROM ORDERS WHERE order_id = cancel_order.order_id;
7
8    DBMS_OUTPUT.PUT_LINE('Order ' || order_id || ' has been cancelled and its associated order details and payment records have been deleted.');
9  EXCEPTION
10   WHEN OTHERS THEN
11     DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
12 END;
13
14
15   declare
16 n number := 11;
17 Begin
18 cancel_order(n);
19 DBMS_OUTPUT.PUT_LINE('Order cancelled');
20 end;
```

Results | Explain | Describe | Saved SQL | History

```
Order 11 has been cancelled and its associated order details and payment records have been deleted.
Order cancelled

Statement processed.
```

0.02 seconds

nnnurda | thevarp18 | en

Buyer cancels his order

TRIGGER

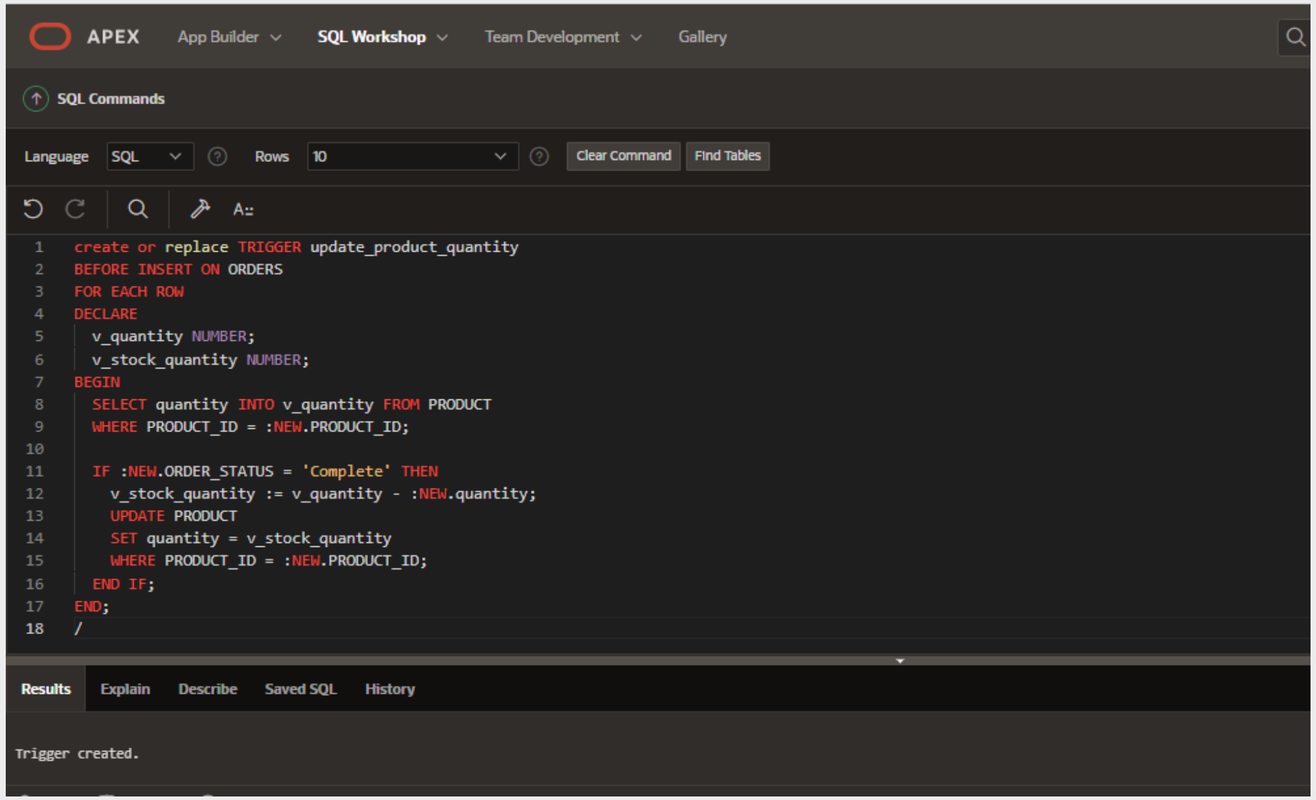
```
1  CREATE OR REPLACE TRIGGER update_orderdetail
2  BEFORE INSERT ON ORDERDETAIL
3  FOR EACH ROW
4  DECLARE
5  v_price_order NUMBER;
6  v_cost NUMBER;
7
8  BEGIN
9  SELECT P.PRODUCT_PRICE INTO v_price_order FROM PRODUCT P
10 JOIN ORDERS ORD ON P.PRODUCT_ID = ORD.PRODUCT_ID
11 WHERE ORD.PRODUCT_ID = :NEW.PRODUCT_ID AND ORD.ORDER_STATUS = 'Complete';
12 v_cost := v_price_order * :NEW.quantity;
13 :NEW.price := v_cost;
14 END;
15
16 insert into ORDERDETAIL(ORDER_ID,PRODUCT_ID,QUANTITY)
17 VALUES(1,1,9);
```

Results | Explain | Describe | Saved SQL | History

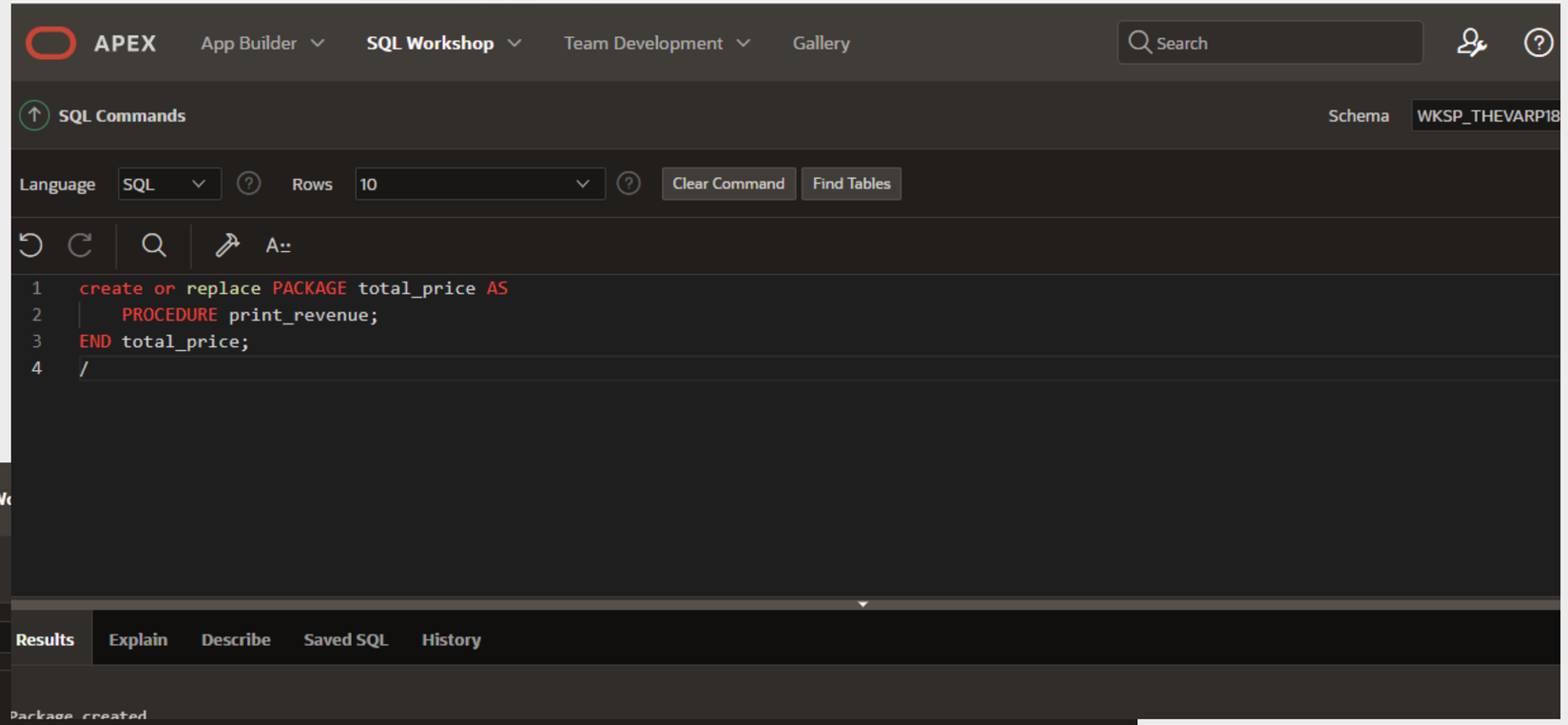
```
1 row(s) inserted.

0.15 seconds
```

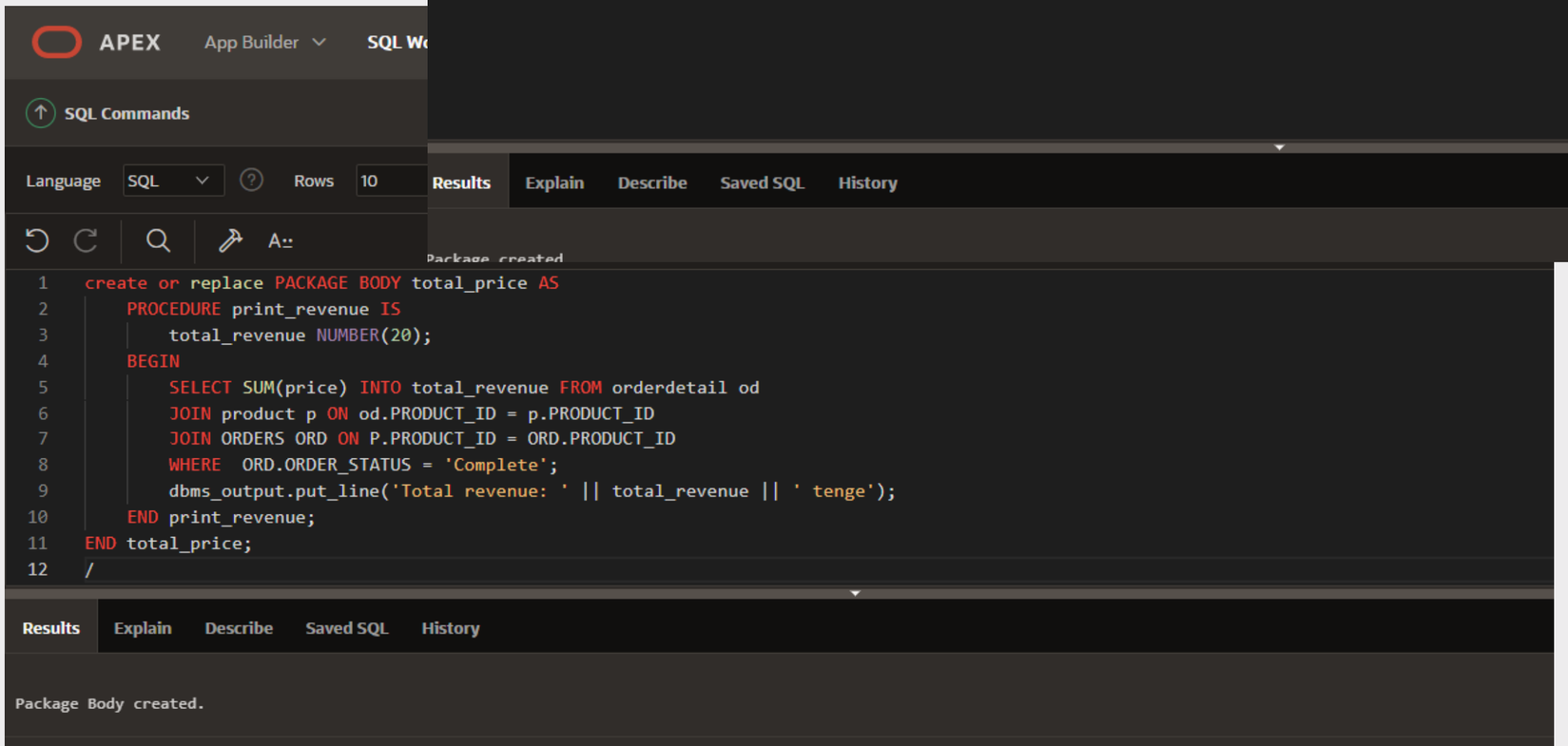
nnnurda | thevarp18 | en

TRIGGER


APEX
App Builder
SQL Workshop
Team Development
Gallery
SQL Commands
Language
SQL
Rows
10
Clear Command
Find Tables
A::
1 create or replace TRIGGER update_product_quantity
2 BEFORE INSERT ON ORDERS
3 FOR EACH ROW
4 DECLARE
5 v_quantity NUMBER;
6 v_stock_quantity NUMBER;
7 BEGIN
8 SELECT quantity INTO v_quantity FROM PRODUCT
9 WHERE PRODUCT_ID = :NEW.PRODUCT_ID;
10
11 IF :NEW.ORDER_STATUS = 'Complete' THEN
12 v_stock_quantity := v_quantity - :NEW.quantity;
13 UPDATE PRODUCT
14 SET quantity = v_stock_quantity
15 WHERE PRODUCT_ID = :NEW.PRODUCT_ID;
16 END IF;
17 END;
18 /
Results
Explain
Describe
Saved SQL
History
Trigger created.

PACKAGE


APEX
App Builder
SQL Workshop
Team Development
Gallery
Search
SQL Commands
Schema
WKSP_THEVARP18
Language
SQL
Rows
10
Clear Command
Find Tables
1 create or replace PACKAGE total_price AS
2 PROCEDURE print_revenue;
3 END total_price;
4 /
Results
Explain
Describe
Saved SQL
History
Package created.


APEX
App Builder
SQL Commands
Language
SQL
Rows
10
1 create or replace PACKAGE BODY total_price AS
2 PROCEDURE print_revenue IS
3 total_revenue NUMBER(20);
4 BEGIN
5 SELECT SUM(price) INTO total_revenue FROM orderdetail od
6 JOIN product p ON od.PRODUCT_ID = p.PRODUCT_ID
7 JOIN ORDERS ORD ON P.PRODUCT_ID = ORD.PRODUCT_ID
8 WHERE ORD.ORDER_STATUS = 'Complete';
9 dbms_output.put_line('Total revenue: ' || total_revenue || ' tenge');
10 END print_revenue;
11 END total_price;
12 /
Results
Explain
Describe
Saved SQL
History
Package Body created.

FUNCTION

```
1   create or replace FUNCTION totalBuyers
2   RETURN number IS
3      total number := 0;
4   BEGIN
5      SELECT count(*) into total
6      FROM buyer;
7
8      RETURN total;
9   END;
10  /
11
12  DECLARE
13     n number;
14  BEGIN
15     n := totalBuyers();
16     dbms_output.put_line('Total no. of Buyers: ' || n);
17  END;
```

Results | Explain | Describe | Saved SQL | History

```
Total no. of Buyers: 52

Statement processed.
```

nnnurda | thevarp18 | en | Copyright © 1999, 2023, Oracle and/

Total number of buyers

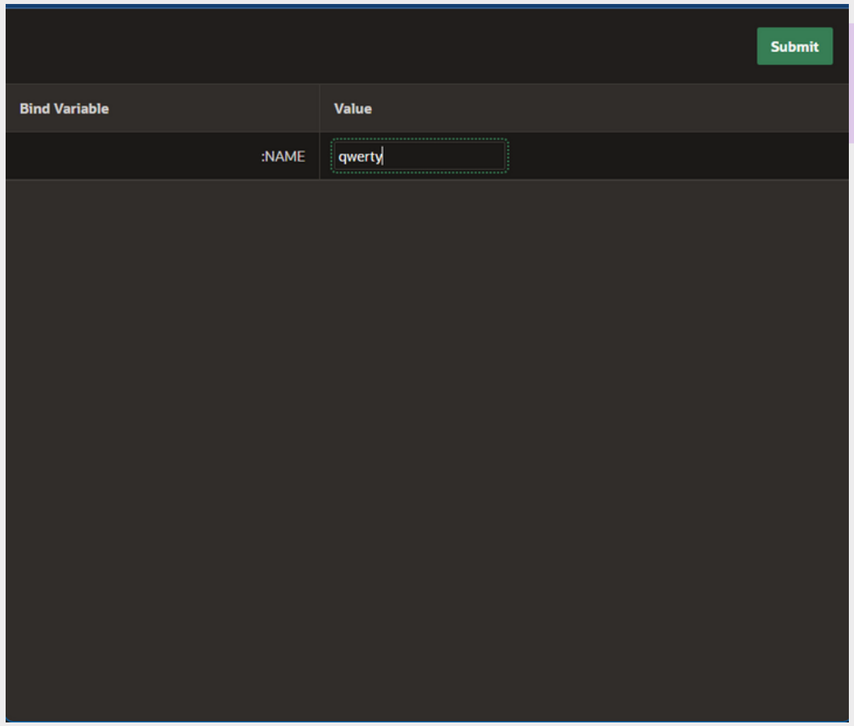
PROCEDURE

```sql
create or replace PROCEDURE print_product_info(p_product_name IN PRODUCT.product_name%TYPE)
IS
    least_five_exception exception;
    v_product_id PRODUCT.product_id%TYPE;
    v_product_price PRODUCT.product_price%TYPE;
    v_product_description PRODUCT.product_description%TYPE;
    v_product_quantity PRODUCT.quantity%type;
BEGIN
    IF LENGTH(p_product_name) < 5 THEN
        RAISE least_five_exception;
    END IF;

    SELECT product_id, product_price, product_description
    INTO v_product_id, v_product_price, v_product_description
    FROM PRODUCT
    WHERE product_name = p_product_name;

    DBMS_OUTPUT.PUT_LINE('Product ID: ' || v_product_id);
    DBMS_OUTPUT.PUT_LINE('Product Name: ' || p_product_name);
    DBMS_OUTPUT.PUT_LINE('Product Price: ' || v_product_price);
    DBMS_OUTPUT.PUT_LINE('Product Description: ' || v_product_description);
EXCEPTION
    WHEN least_five_exception THEN
        DBMS_OUTPUT.PUT_LINE('Product name should be at least 5 characters.');
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Product not found.');
END;
/

Declare
Begin
print_product_info(:name);
End;
```

Results | Explain | Describe | Saved SQL | History

```
Product not found.

Statement processed.
```

nnnurda thevarp18 en Copyrig



FUNCTION

```sql
create or replace FUNCTION get_total_orders_for_buyer(buyer IN NUMBER)
RETURN NUMBER
IS
  total_orders NUMBER;
BEGIN
  SELECT COUNT(*) INTO total_orders FROM ORDERS WHERE BUYER_ID = buyer;
  RETURN total_orders;
  IF SQL%ROWCOUNT=0 then
    raise NO_DATA_FOUND;
    END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No orders found for buyer with ID ' || buyer);
    RETURN 0;
END;
/

Declare
Begin
update_buyer_phone(4, 7747589326);
End;
```

Results | Explain | Describe | Saved SQL | History

```
Buyer phone number updated successfully.

Statement processed.
```

nnnurda thevarp18 en

THANK YOU
FOR ATTENTION!