**DATABASE MANAGEMENT SYSTEMS II**

***Team members:***

Akbayan Berdikhanova 210103220

Tomiris Akyl 210103212

Assel Kemalova 210103368

Gulnaz Yekpin 210103067

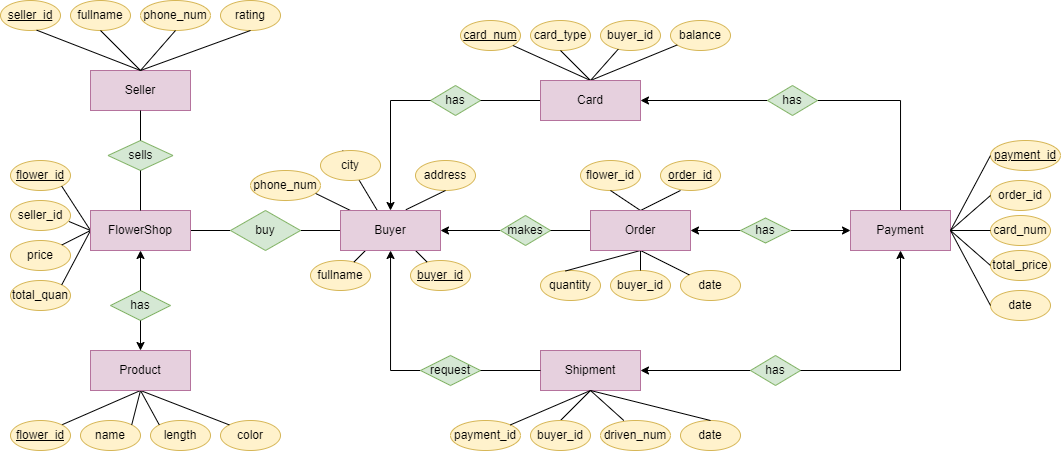
Kumissay Amantay 210103312

**DESIGN PHASE: Flower Shop**

**Description:**

A database is an ordered set of structured information or data that is usually stored electronically in a computer system.

Our project is about an online flower shop. You can order flowers from us through the seller, and with online payment, flowers are delivered to the designated address. Our system performs billing. The main task of designing is to create a software product that reflects all the information about our flower project. The process of buying flowers in an online store. The purchase process begins from the moment the buyer makes an order.

**ER Diagram**

**NORMALIZATION**

**Buyer(buyer\_ID, city, address, fullname, phone\_num)**

Primary Key: buyer\_ID

Functional Dependencies: buyer\_ID → city, address, fullname, phone\_num

This table is already in 3NF.

**Seller(sellerID, fullname, phone\_num, rating)**

Primary Key: sellerID

Functional Dependencies: sellerID → fullname, phone\_num, rating

This table is already in 3NF.

**FlowerShop(sellerID, flowerID, total\_quan, price)**

Primary Key: (sellerID, flowerID)

Functional Dependencies: sellerID, flowerID → total\_quan, price

This table is already in 3NF.

**Product(FlowerId,name,length,color)**

Primary Key: FlowerId

Functional Dependencies: FlowerId → name, length, color

This table is already in 3NF.

**Order(orderID,flowerID, buyerID, quantity, date)**

Primary Key: orderID

Functional Dependencies: orderID → flowerID, buyerID, quantity, date

This table is already in 3NF.

**Payment(paymentID, orderID, total\_price, date, card\_number)**

Primary Key: paymentID

Functional Dependencies: paymentID → orderID, total\_price, date, card\_number

This table is already in 3NF.

**Card(card\_number, buyerId, card\_type, balance)**

Primary Key: card\_number

Functional Dependencies: card\_number → buyerId, card\_type, balance

This table is already in 3NF.

**Shipment(payment\_ID, buyerID, date, driver\_num)**

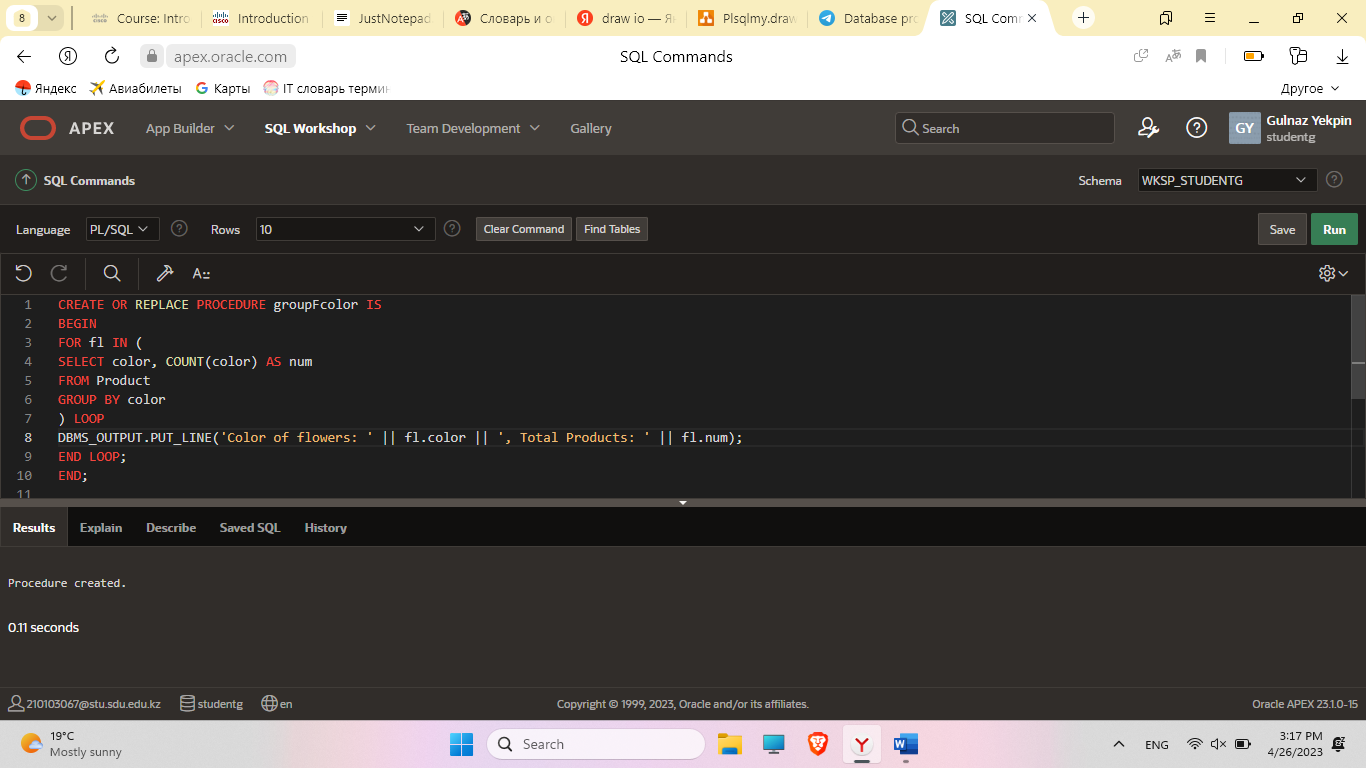
Primary Key: payment\_ID

Functional Dependencies: payment\_ID → buyerID, date, driver\_num

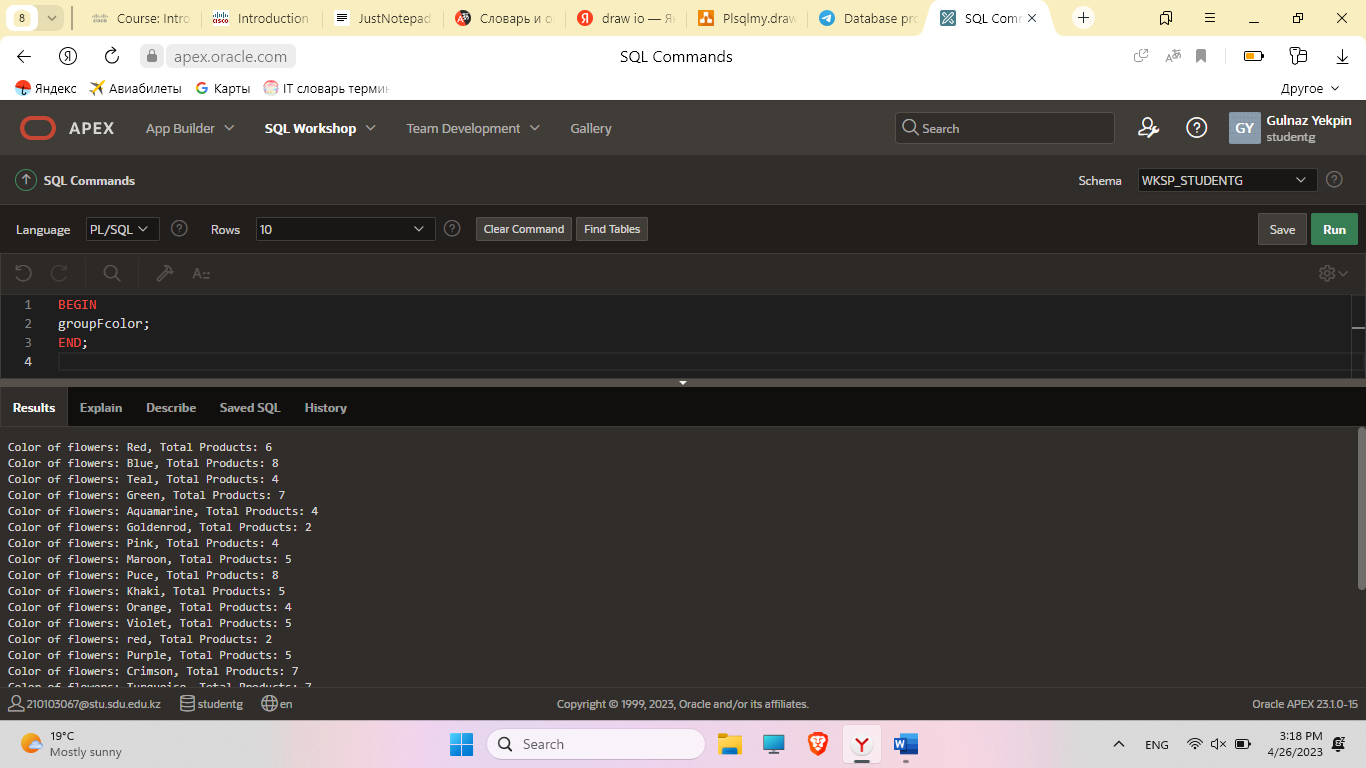
This table is already in 3NF.

1. Procedure that groups flower in Product table by color:

CREATE OR REPLACE PROCEDURE groupFcolor IS  
BEGIN  
FOR fl IN (  
SELECT color, COUNT(color) AS num  
FROM Product  
GROUP BY color  
) LOOP  
DBMS\_OUTPUT.PUT\_LINE('Color of flowers: '  fl.color  ', Total Products: ' || fl.num);  
END LOOP;  
END;

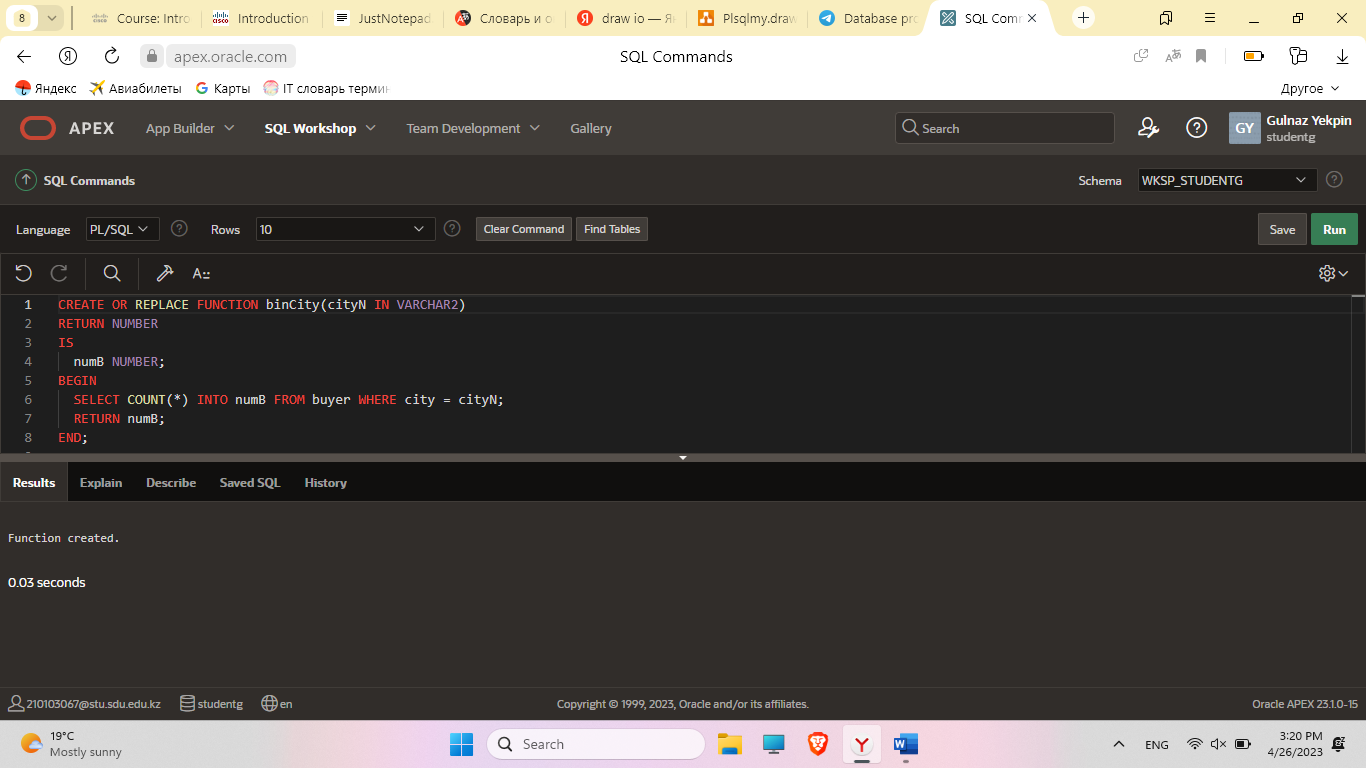


BEGIN  
groupFcolor;  
END;

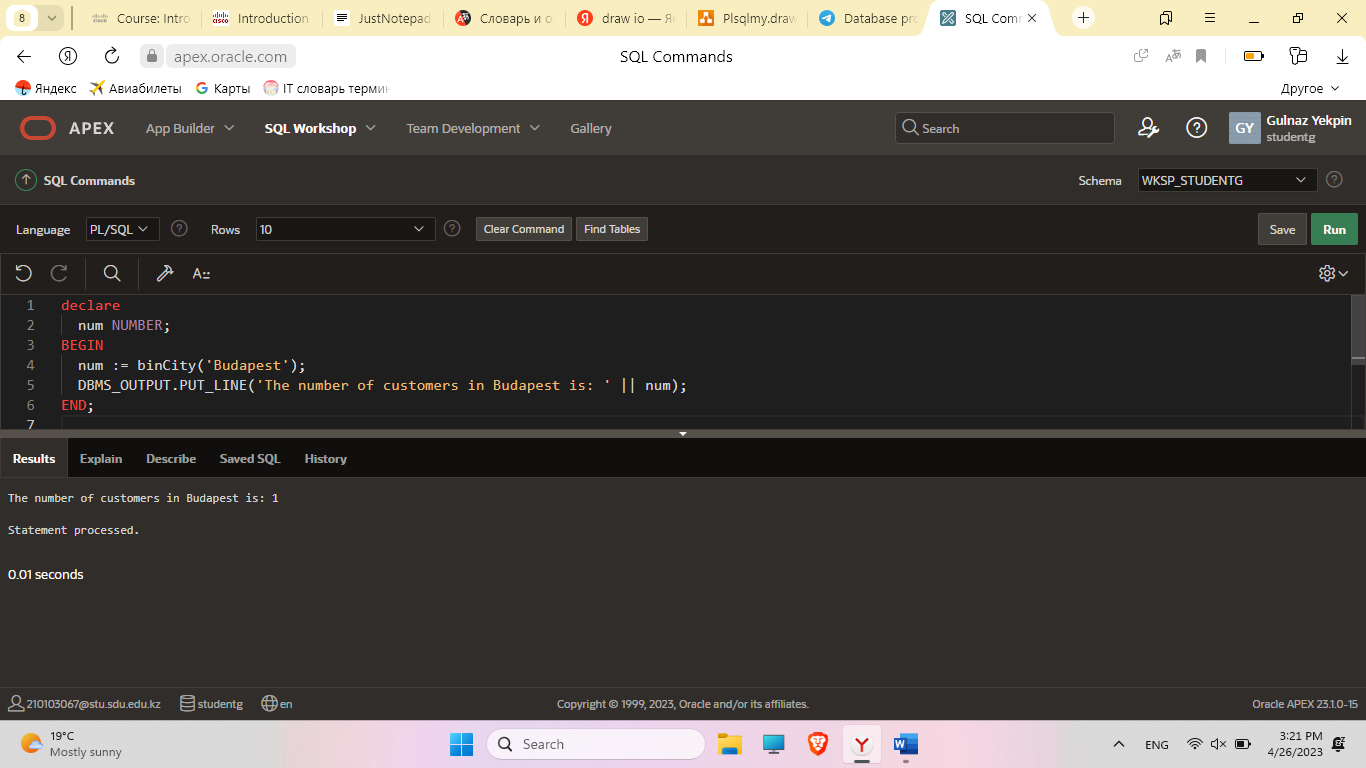


1. Function which counts the number of buyers in one city:

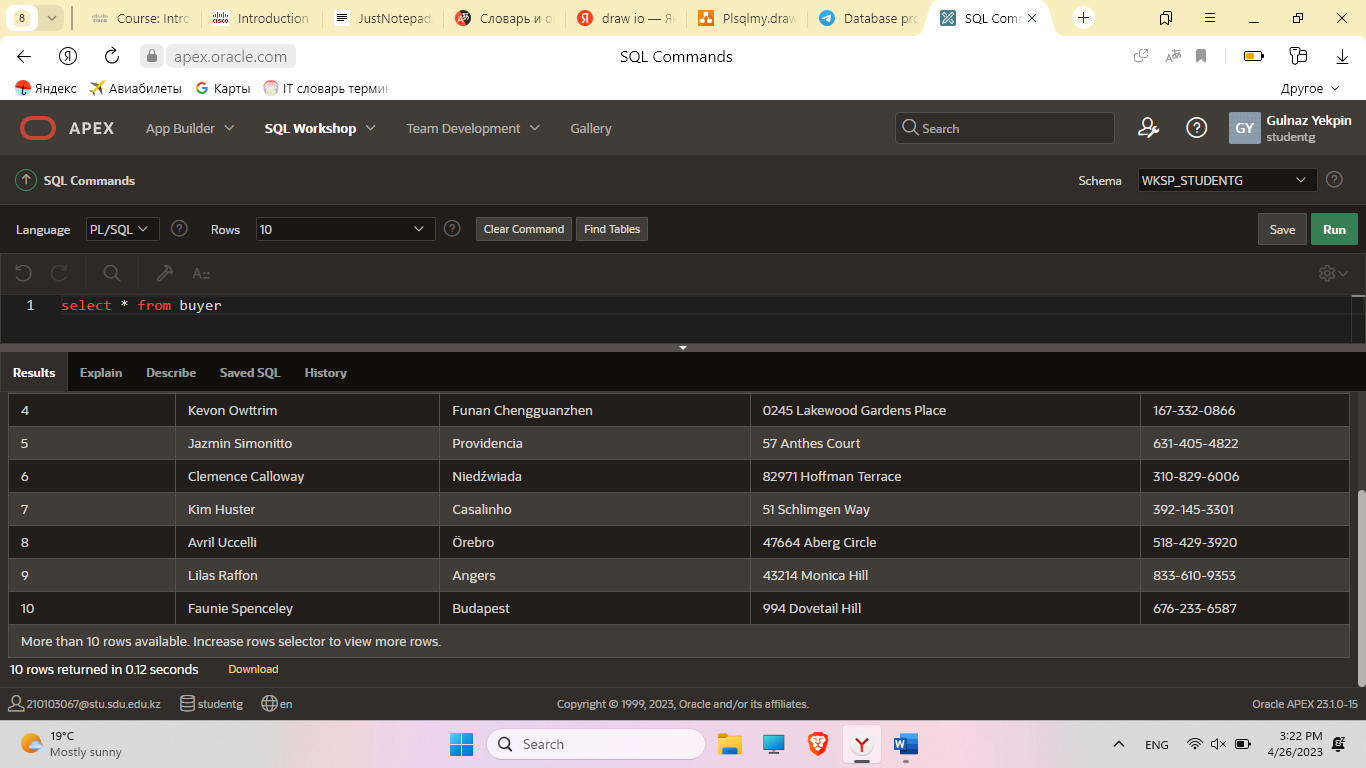
CREATE OR REPLACE FUNCTION binCity(cityN IN VARCHAR2)  
RETURN NUMBER  
IS  
  numB NUMBER;  
BEGIN  
  SELECT COUNT(\*) INTO numB FROM buyer WHERE city = cityN;  
  RETURN numB;  
END;



declare  
  num NUMBER;  
BEGIN  
  num := binCity('Budapest');  
  DBMS\_OUTPUT.PUT\_LINE('The number of customers in Budapest is: ' || num);  
END;

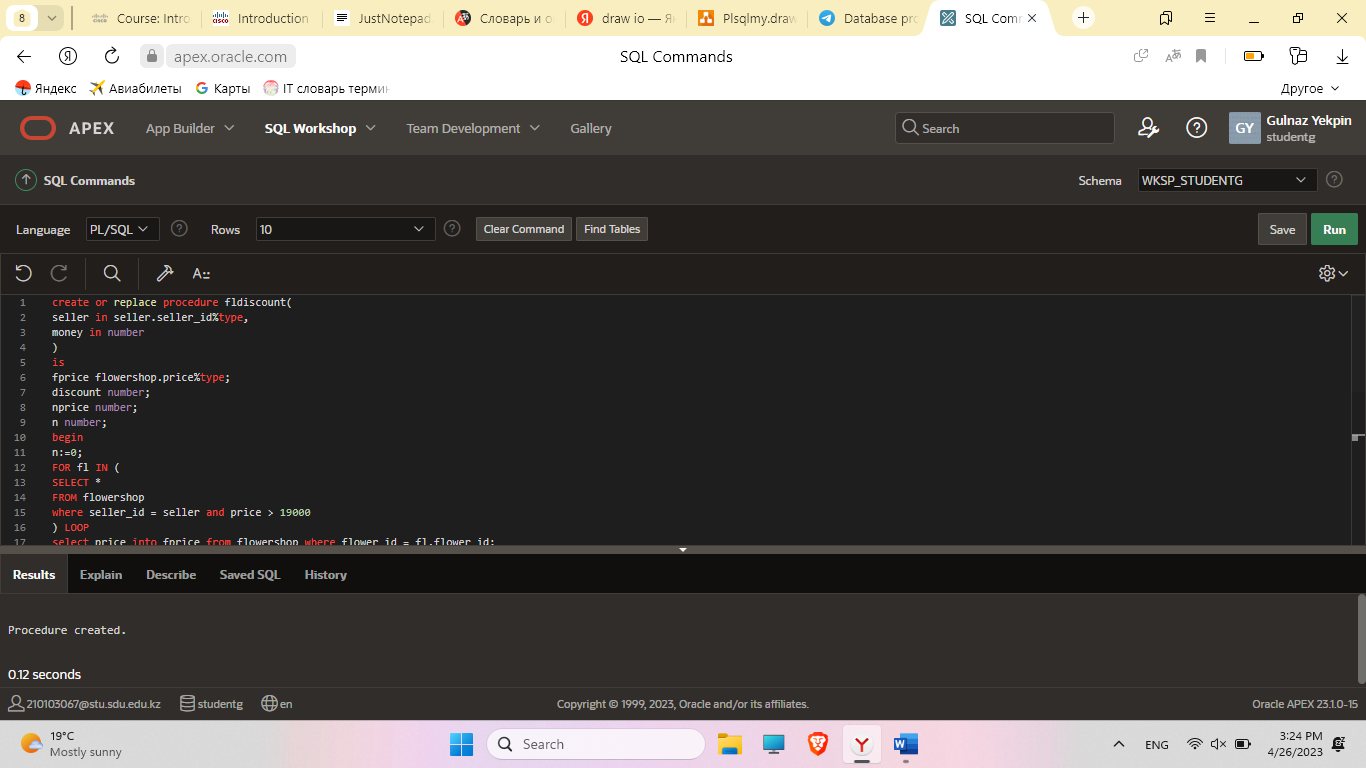


select \* from buyer

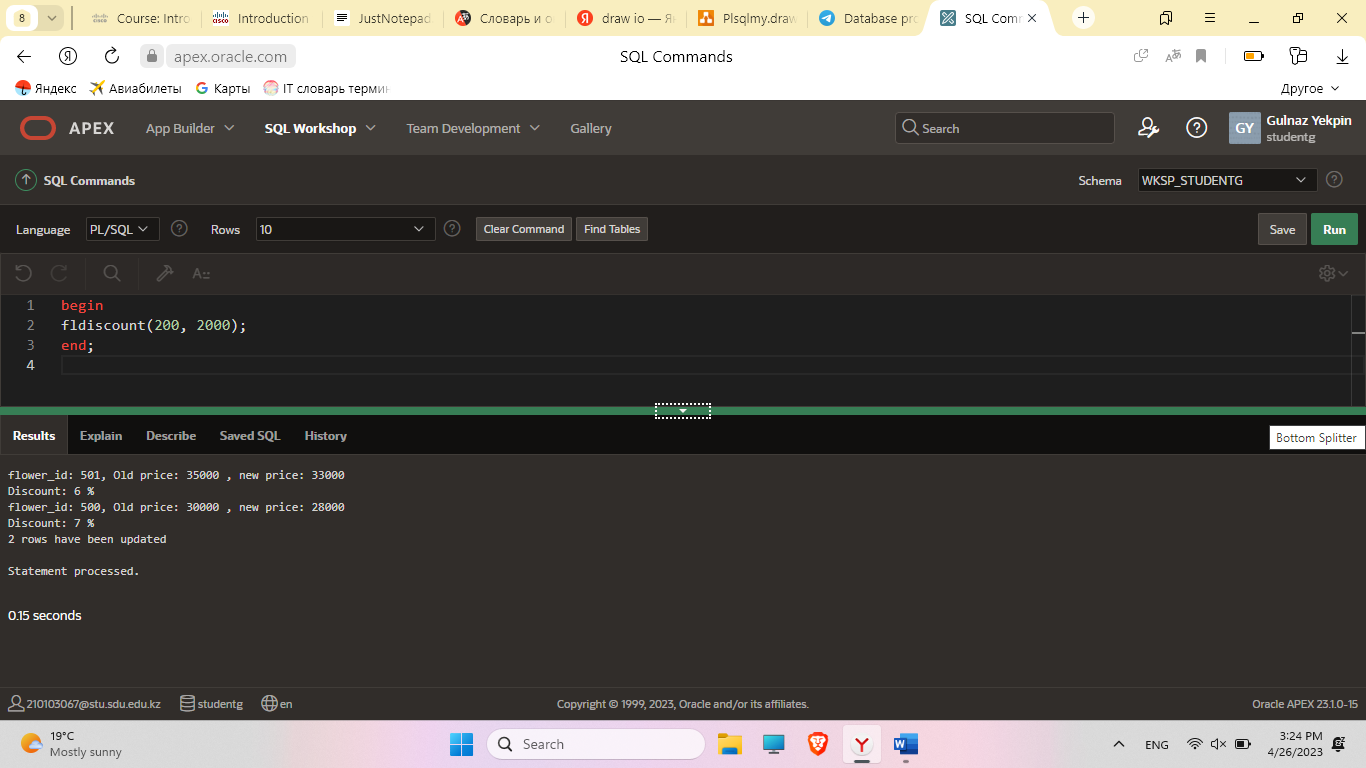


1. Procedure which uses SQL%ROWCOUNT to determine the number of rows affected

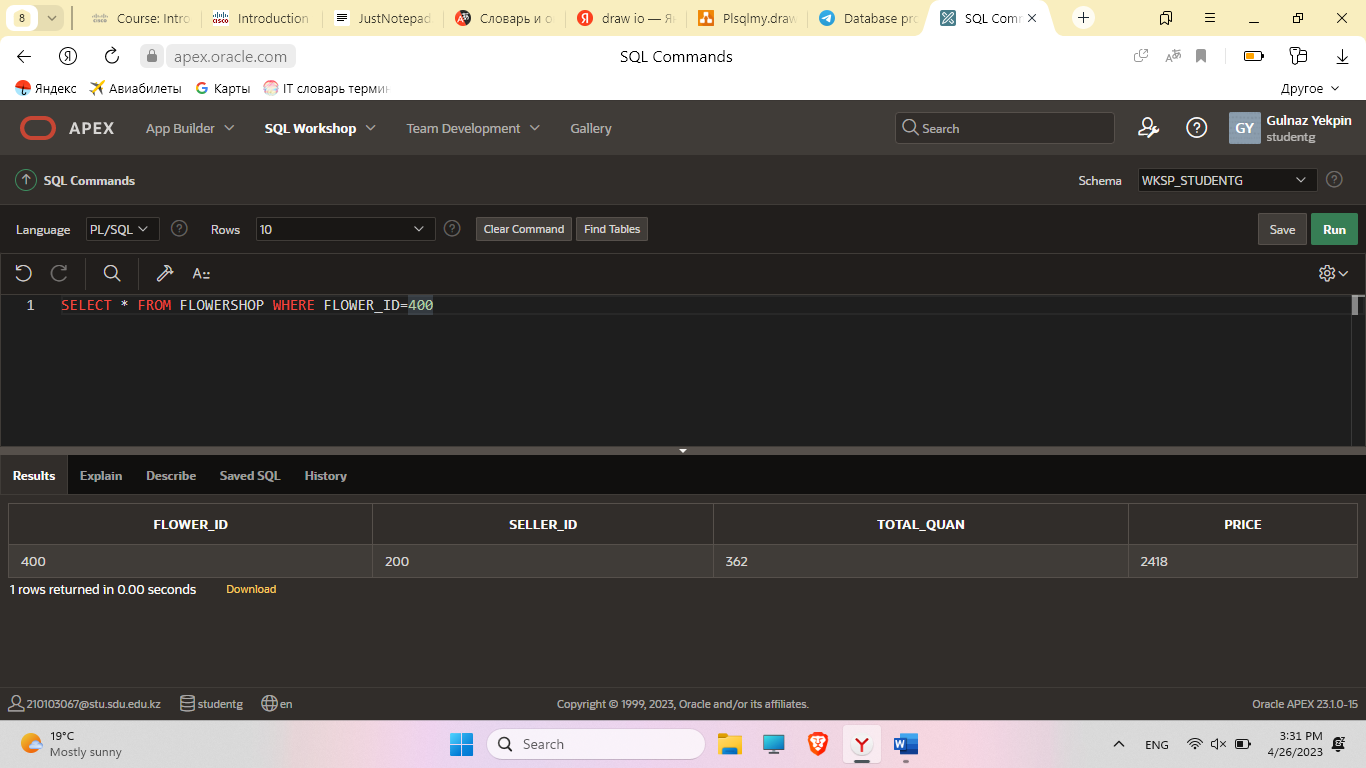
create or replace procedure fldiscount(  
seller in seller.seller\_id%type,  
money in number  
)  
is  
fprice flowershop.price%type;  
discount number;  
nprice number;  
n number;  
begin  
n:=0;  
FOR fl IN (  
SELECT \*  
FROM flowershop  
where seller\_id = seller and price > 19000  
) LOOP  
select price into fprice from flowershop where flower\_id = fl.flower\_id;  
nprice := fprice - money;  
update flowershop set price = nprice where flower\_id = fl.flower\_id;  
n := n + sql%rowcount;  
discount := 100 - round(100 \* (nprice/fprice));  
DBMS\_OUTPUT.PUT\_LINE('flower\_id: '  fl.flower\_id  ', Old price: '  fprice  ' , new price: '  nprice) ;  
DBMS\_OUTPUT.PUT\_LINE('Discount: '  discount  ' %');  
END LOOP;  
DBMS\_OUTPUT.PUT\_LINE(n  ' rows have been updated');  
end;

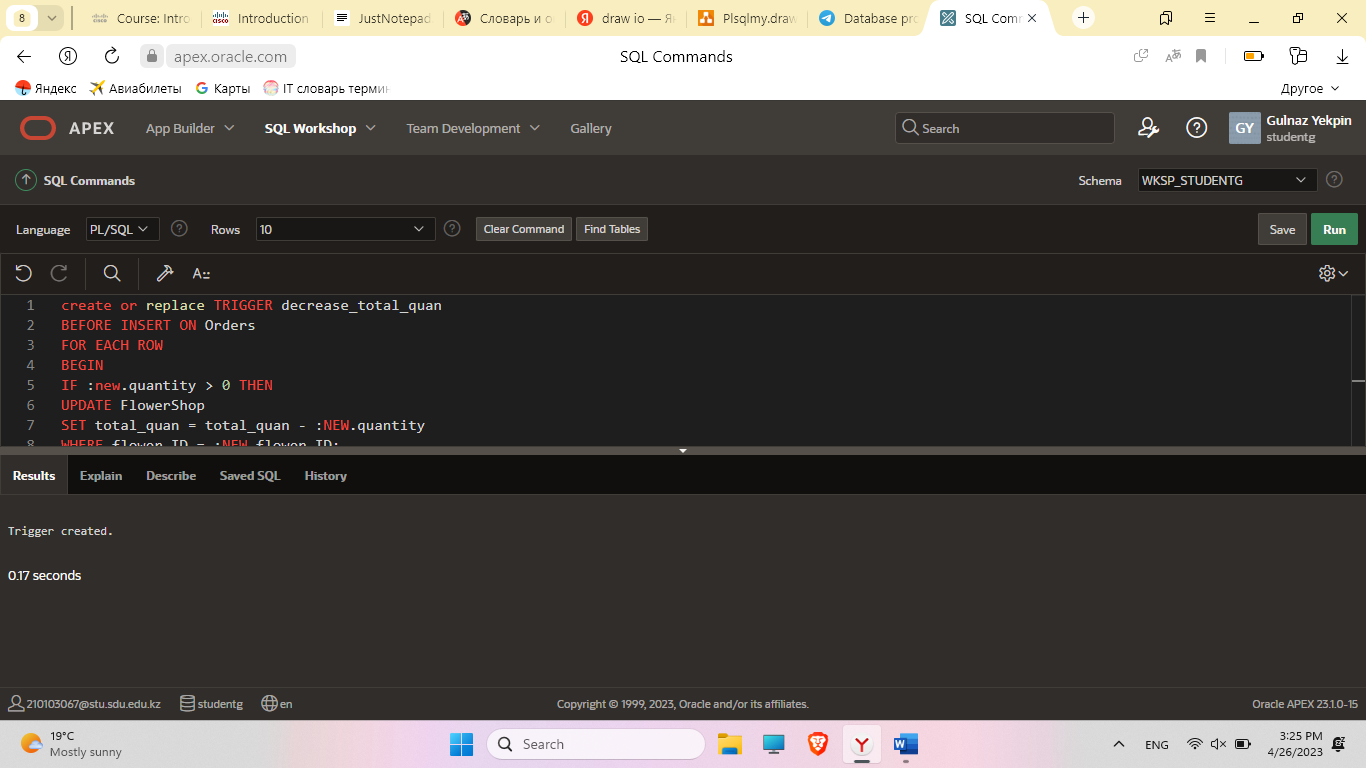


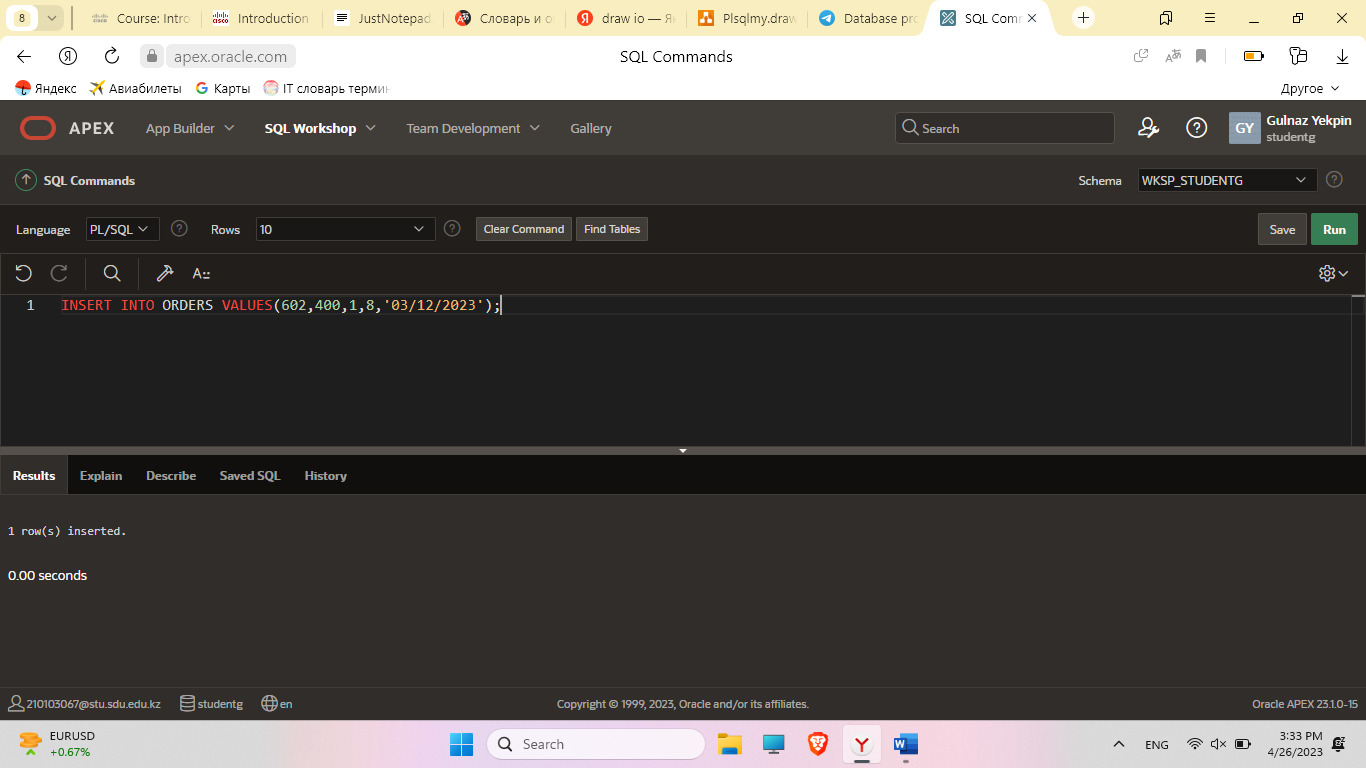
begin  
fldiscount(200, 2000);  
end;

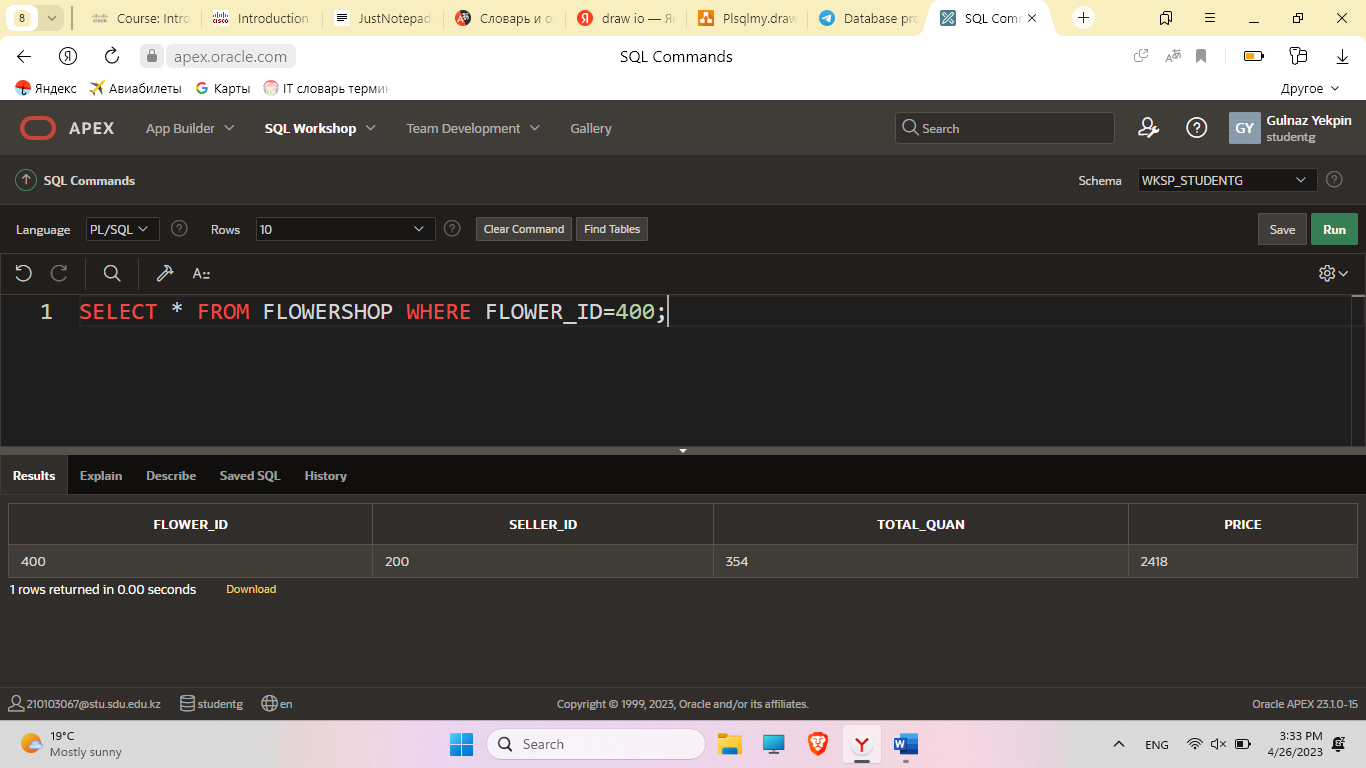


1. Trigger which decreases quantity of flower in flowershop table if buyer made order to that flower:  
   create or replace TRIGGER decrease\_total\_quan  
   BEFORE INSERT ON Orders  
   FOR EACH ROW  
   BEGIN  
   IF :new.quantity > 0 THEN  
   UPDATE FlowerShop  
   SET total\_quan = total\_quan - :NEW.quantity  
   WHERE flower\_ID = :NEW.flower\_ID;  
   end if;  
   END;









1. Create a trigger before insert on any entity which will show the current number of rows in the table

create or replace trigger paymenttrig before insert on payment

for each row declare money number;

pay exception;

begin select balance into money from card

where card\_number = :new.card\_number;

if money < :new.total\_price then raise pay;

else update card set balance = balance - :new.total\_price

where card\_number = :new.card\_number;

end if;

exception when pay then

dbms\_output.put\_line('There are not enough funds on your card to pay!!!');

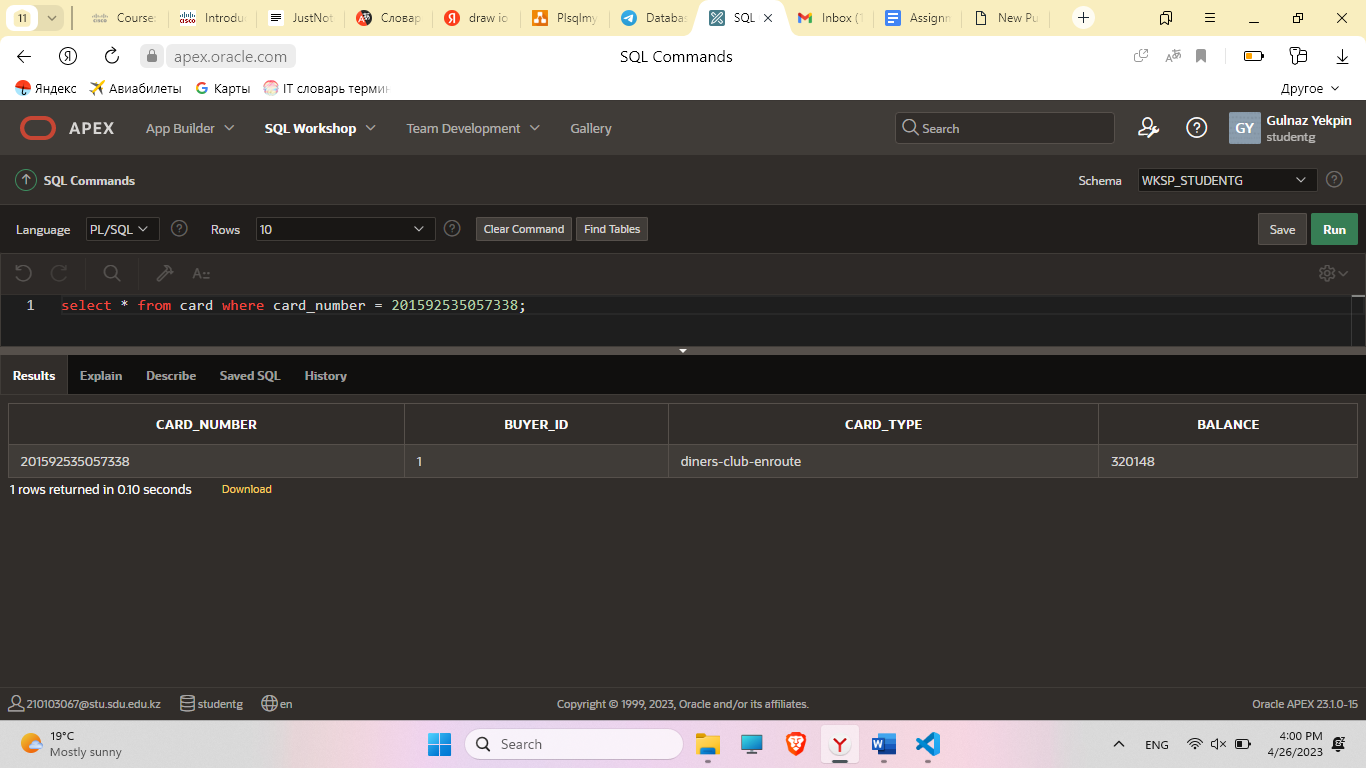
end;

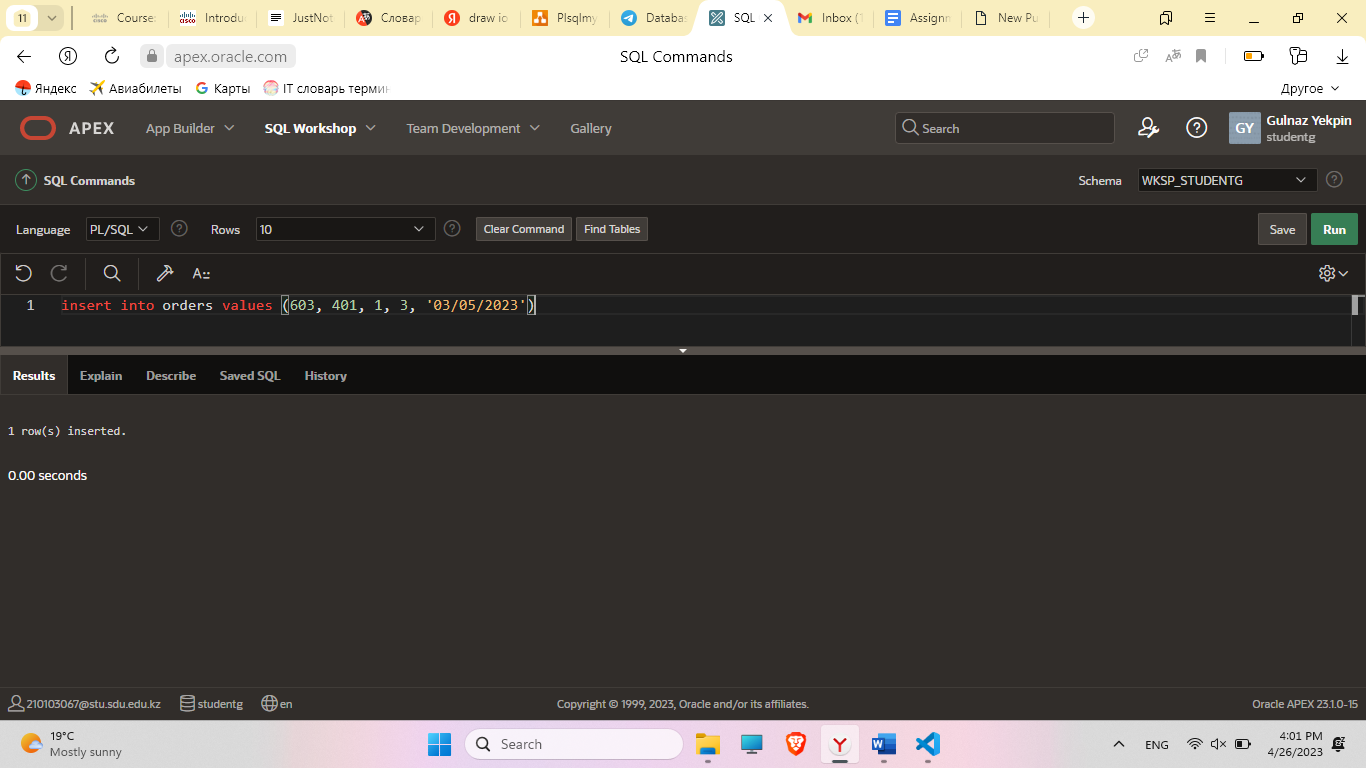
select \* from card where card\_number = 201592535057338;

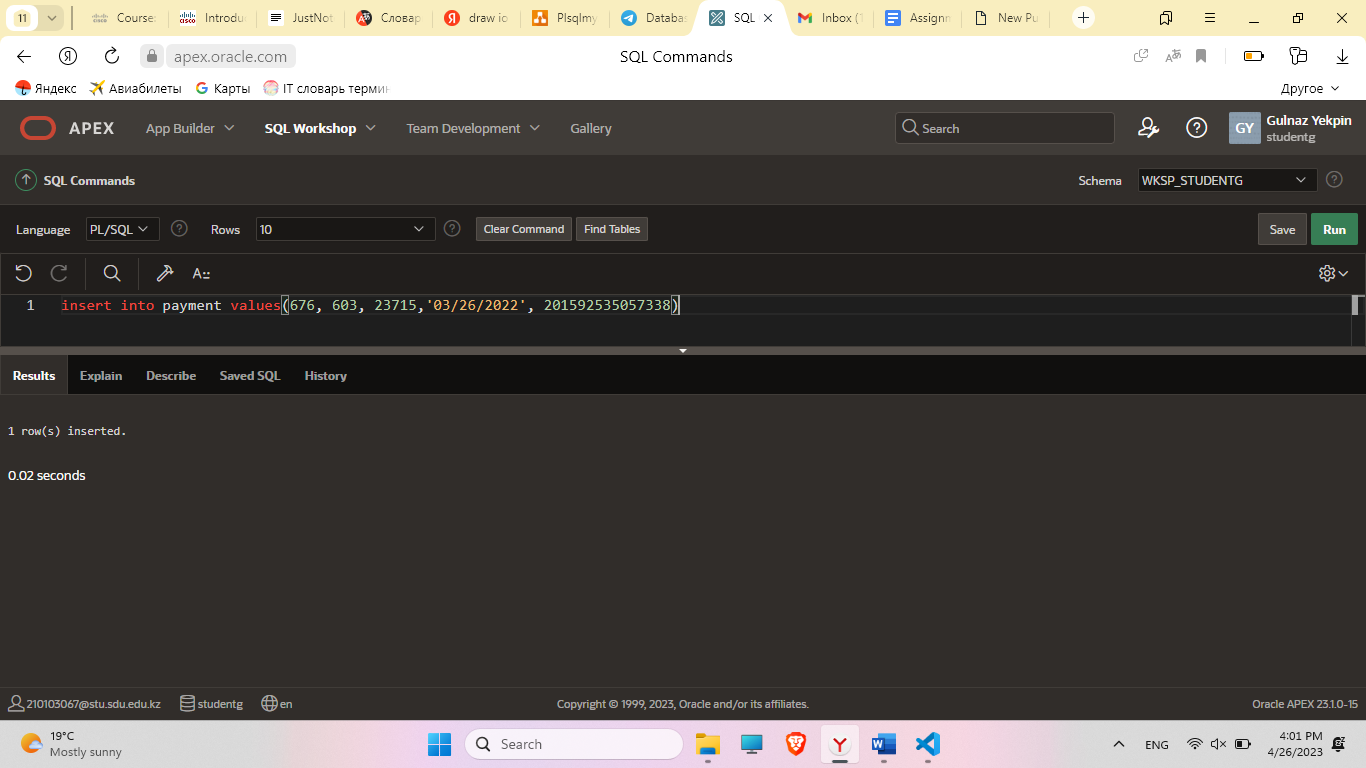
insert into orders values (603, 401, 1, 3, '03/05/2023')

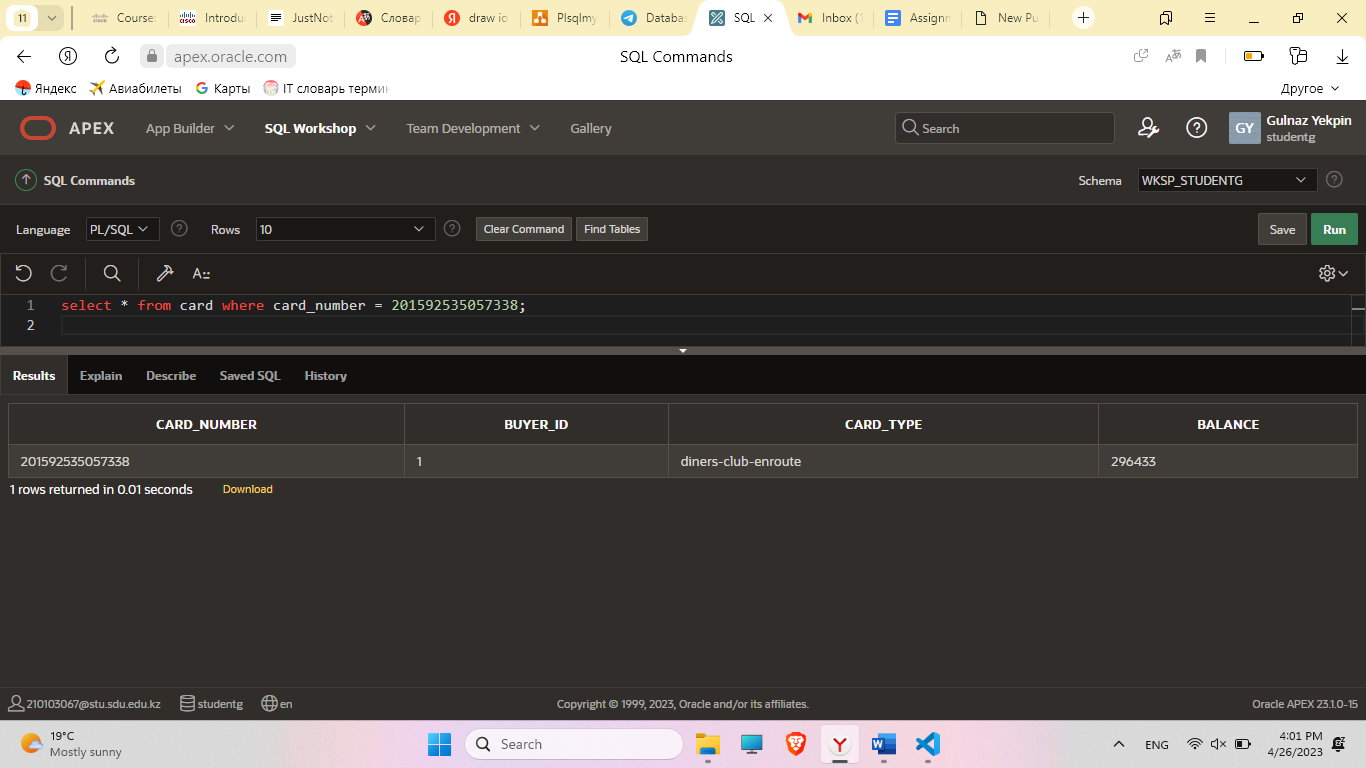
insert into payment values(676, 603, 23715,'03/26/2022', 201592535057338)

select \* from card where card\_number = 201592535057338;









*--Exception--*

select \* from card where card\_number = 271416;  
insert into orders values (604, 401, 1, 3, '03/05/2023')  
insert into payment values(677, 604, 23715,'03/26/2022',271416 )  
select \* from card where card\_number = 271416;

