

WOMEN'S CLOTHING STORE

210103397 – Kozhamuratova Aizhan
210103469 – Kuralay Mukhtar
210103159 – Bergen Saule
210103190 – Ayana Bakytzhanova
210103038 – Ayala Narbekova

Description:

i) Introduction and database description.

Our project is designed for an online women's clothing store. The "women's clothing store" database contains information about sellers, buyers and couriers.

At the beginning, when a user registers, he chooses his category, to which one of the above subclasses belongs. Each type of user specifies its own address and card. Users register only by phone number.

In our database, a seller can sell many products. Products have own pid, sizes, category, price and sid which connects with seller.

Each customer has a shopping cart, where you can store different products that you want to buy.

Transactions will take correspond the shopping cart and stores all the information of the product that you are going to buy. After that, the information is delivered to the courier.

ii) What functions should the system perform?

- Allowing users to register an account with their personal information.
- Manage profile information, such as updating their name, address, and card details.
- To place an order for a product, specify your address for delivery and card details for payment.
- Allowing sellers to list their products for sale, providing product details such as category, price and size.
- Buyers can search for products available from sellers in the system.
- Find out the registration date for each user.

iii) Who are the end users?

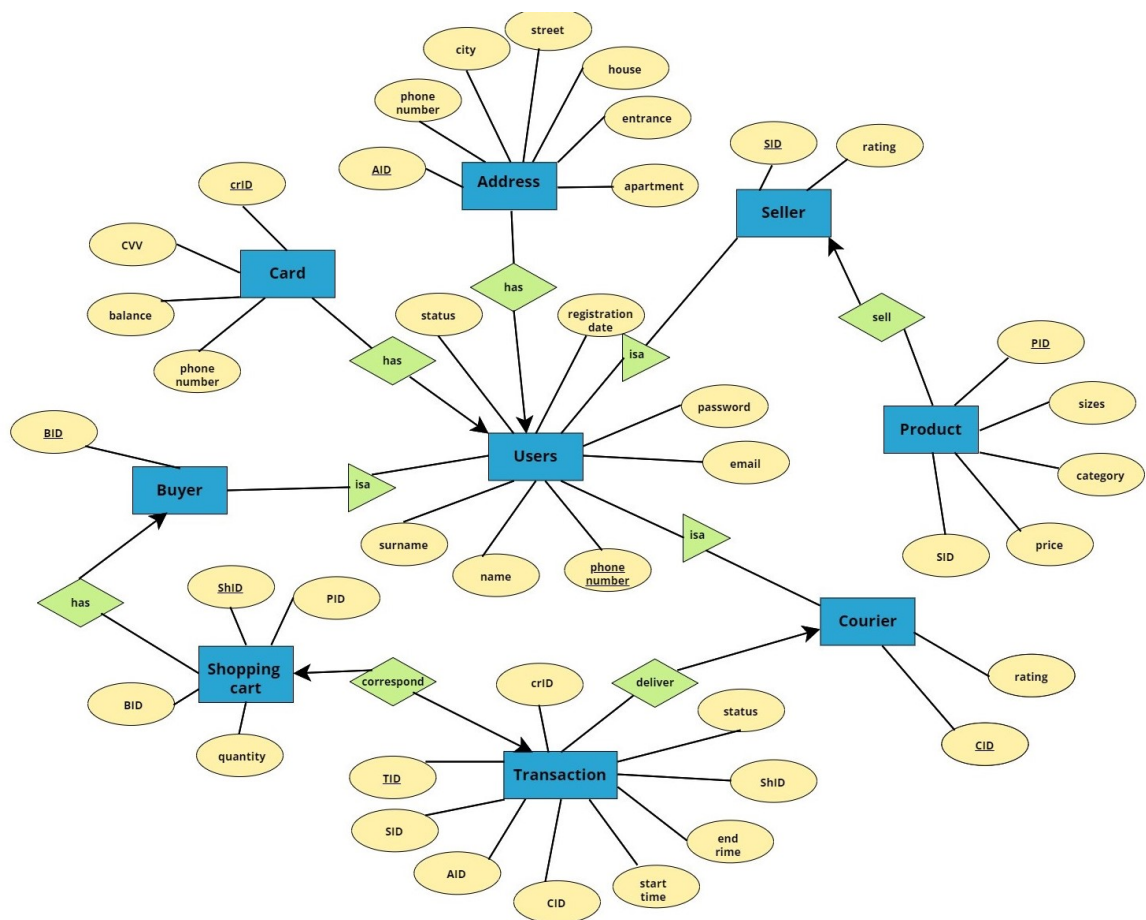
Sellers, Buyers and Couriers.

iv) Where did we get the idea for this project?

v) How will data obsolescence be handled?

ENTITY RELATIONSHIP DESIGN

The



"users" entity has the attributes "phone_number", "name", "surname", "email", "password", "registration_date" and "status". The user registers only by phone number. So, this is the key attribute. The status can be buyer, seller or courier. Users have subclasses: "buyer", "seller" and "courier".

In a one-to-many relationship between "user" and "card", it means that one user can have multiple cards, while each card belongs to only one user. "card" has the attributes crID, CVV, balance and phone_number.

In a one-to-many relationship between "user" and "address", it means that one user can have multiple addresses, while each address ID belongs to only one user. "address" has the attributes AID, phone_number, city, street, house, entrance and apartment.

In this one-to-many relationship between "seller" and "product", a seller (identified by SID) can sell multiple products, while each product (identified by PID) is associated with only one seller. The "seller" entity has a "raiting" attribute, which could represent the seller's rating. The "product" entity has attributes such as "size", "category", and "price" (which could represent the current price of the

product). The "category" attribute defines the type of clothing (for example, skirt, jacket, jeans, etc.)

In this one-to-many relationship between "buyer" and "shopping cart", a buyer (identified by BID) can have multiple shopping carts, and each shopping cart (identified by ShID) is uniquely associated with the buyer's account and can contain product ID and quantity of product.

In this one-to-one relationship between "shopping cart" and "transaction", each shopping cart(identified by ShID) is linked to a single transaction(identified by TID), and vice versa. Once the buyer proceeds to the checkout process and completes the purchase, the shopping cart is converted into a transaction, capturing all the relevant details of the purchase. The "transaction" entity has attributes such as "start time", "end time", "crID"(card ID), ShID, AID(address ID), SID(seller ID), status and "CID" (courier ID). The status of transaction can be declined, approved or pending.

In this many-to-one relationship between "transaction" and "courier", multiple transactions (identified by TID) can be associated with a single courier (identified by CID), who can be rated by users. The "Courier" entity represents individual couriers with attributes such as "CID" (which serves as the primary key for the courier) and "rating" (which represents the rating given by users).

NORMALIZATION

Table User

Key: phone_number

phone_number -> password, email, name, surname, status, registration_date

1NF: Each user will register with a phone number and this will be the primary key. And each user can have only one password, email, name, surname, status, registration_date.

2NF: All nonkey attributes are functionally dependent on the entire phone_number, so that's 2NF.

3NF: In the existing FD, the left hand side is the key, so it's 3NF.

Table Seller

Key: SID

Foreign Key: phone_number

phone_number -> SID, rating

SID -> phone_number, rating

1NF: Each seller can have only one phone_number, SID, rating.

2NF: Nonkey attribute rating is functionally dependent on the entire SID or phone_number.

3NF: In the existing FDs, the left hand side is the key or its right hand side is a key.

Table Buyer

Key: BID

Foreign Key: phone_number

phone_number -> BID

BID -> phone_number

1NF: Each buyer can have only one phone_number, BID

2NF: in 1NF and does not have nonkey attributes.

3NF: In the existing FDs, the left hand side is the key or its right hand side is a key.

Table Courier

Key: CID

Foreign Key: phone_number

phone_number -> CID, rating

CID -> phone_number, rating

1NF: Each courier can have only one phone_number, CID, rating.

2NF: Nonkey attribute rating is functionally dependent on the entire CID or phone_number.

3NF: In the existing FDs, the left hand side is the key or its right hand side is a key.

Table Card

Key: crID

crID -> CVV, balance, phone_number

1NF: Each card can have only one CVV, balance, phone_number

2NF: Nonkey attributes are functionally dependent on the entire crID.

3NF: In the existing FD, the left hand side is the key.

Table Address

Key: AID

AID -> phone_number, city, street, house, entrance, apartment

1NF: Each address ID can have only one phone_number, city, street, house, entrance, apartment.

2NF: Nonkey attributes are functionally dependent on the entire AID.

3NF: In the existing FD, the left hand side is the key.

Table Product

Key: PID

Foreign Key: SID

PID -> size, category, price, SID

1NF: Each address product can have only one category, price, SID.

2NF: Nonkey attributes are functionally dependent on the entire PID.

3NF: In the existing FD, the left hand side is the key.

Table Shopping_cart

Key: ShID

ShID -> PID, BID, quantity

1NF: Each shopping cart can have only one product ID, buyer and quantity.

2NF: Nonkey attributes are functionally dependent on the entire ShID.

3NF: In the existing FD, the left hand side is the key.

Table Transaction

Key: TID

TID -> SID, status, ShID, CID, AID, start_time, end_time, crID

1NF: Each transaction can have only one SID, status, ShID, CID, AID, start_time, end_time, crID.

2NF: Nonkey attributes are functionally dependent on the entire TID.

3NF: In the existing FD, the left hand side is the key.

CODING PART

- 1) Create table Users

SQL Commands

apex.oracle.com/pls/apex/r/apex-sql-workshop/sqlcommandprocessor?session=8723122519658

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Schema WKSP_SAUBER

Language PL/SQL Rows 20 Clear Command Find Tables Save Run

```

1 create table Users (
2   phone_number VARCHAR(50) primary key,
3   name VARCHAR(50),
4   surname VARCHAR(50),
5   email VARCHAR(50),
6   password VARCHAR(50),
7   registrationDate DATE,
8   status VARCHAR(7)
9 );

```

Results Explain Describe Saved SQL History

PHONE_NUMBER	NAME	SURNAME	EMAIL	PASSWORD	REGISTRATIONDATE	STATUS
9805572694	Cicely	Carbonell	ccarbonell@apache.org	0dnK2z5t13	10/14/2021	buyer
5151524154	Yetty	Quakley	yquakley@bigcartel.com	QJE3mRO5pNg	10/14/2021	seller
695715268	May	Deerr	mdeerr8@google.it	M8r5r7w	08/03/2021	courier
579119595	Kiersten	Calvey	kcalvey@google.com.hk	b5loVWgeD	08/13/2021	courier
172487788	Myles	McMurray	mcmcmurray@networksolutions.com	L9t5bY	04/26/2021	seller

Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.1.0.15

2) Create table Seller

SQL Commands

apex.oracle.com/pls/apex/r/apex-sql-workshop/sqlcommandprocessor?session=8723122519658

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Schema WKSP_SAUBER

Language PL/SQL Rows 20 Clear Command Find Tables Save Run

```

1 create table Seller (
2   sid INT primary key,
3   rating DECIMAL(2,1)
4 );

```

Results Explain Describe Saved SQL History

SID	RATING
5	4.9
8	4.1
10	1.2
1	2.1
17	3.2
2	4.3
9	4.8
3	2.4

Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.1.0.15

3) Create table Product

SQL Commands

apex.oracle.com/pls/apex/r/apex-sql-workshop/sqlcommandprocessor?session=8723122519658

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Schema WKSP_SAUBER

Language PL/SQL Rows 20 Clear Command Find Tables Save Run

```

1 create table Product (
2   pid INT primary key,
3   sizes VARCHAR(50),
4   category VARCHAR(6),
5   price INT,
6   sid INT,
7   FOREIGN KEY(sid) REFERENCES Seller(sid)
8 );

```

Results Explain Describe Saved SQL History

PID	SIZES	CATEGORY	PRICE	SID
1	XS	Jacket	14632	-
2	M	Skirt	24257	-
6	XS	Shorts	13557	-
10	2XL	Shirt	25990	-
14	M	Shorts	35030	-
20	L	Skirt	34577	-

Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.1.0.15

4) Create table Courier

SQL Commands

apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=8723122519658

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Schema WKSP_SAUBER

Language PL/SQL Rows 20 Clear Command Find Tables Save Run

```

1 create table Courier (
2   cid INT primary key,
3   rating INT
4 );

```

Results Explain Describe Saved SQL History

CID	RATING
8	1
1	3
2	3
14	2
3	5
11	3
13	2
20	2

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.10-15

5) Create table Card

SQL Commands

apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=8723122519658

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Schema WKSP_SAUBER

Language PL/SQL Rows 20 Clear Command Find Tables Save Run

```

1 create table Card (
2   crid INT primary key,
3   cvv int,
4   balance INT,
5   phone_number VARCHAR(50),
6   foreign key (phone_number) references users(phone_number)
7 );

```

Results Explain Describe Saved SQL History

CRID	CVV	BALANCE	PHONE_NUMBER
10	338	94906	2242417582
17	634	95952	5791199395
5	398	99249	6967008136
8	557	160975	9769737264
4	730	26918	5788091415
1	248	190490	9803372694
7	477	176478	4448767807

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.10-15

6) Create table Buyer

SQL Commands

apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=8723122519658

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Schema WKSP_SAUBER

Language PL/SQL Rows 20 Clear Command Find Tables Save Run

```

1 create table Buyer (
2   bid INT primary key,
3   phone_number varchar(50),
4   foreign key (phone_number) references users
5 );

```

Results Explain Describe Saved SQL History

BID	PHONE_NUMBER
19	1724877818
2	4668252897
10	2242417582
1	9803372694
3	1137177958
7	5131524154
8	9769737264
9	6937115268

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.10-15

7) Create table Shopping_cart

SQL Commands

apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=8723122519658

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Schema WKSP_SAUBER

Language PL/SQL Rows 20 Clear Command Find Tables Save Run

```

1 create table Shopping_cart (
2   shid INT primary key,
3   pid INT,
4   bid INT,
5   quantity INT,
6   FOREIGN KEY (pid) REFERENCES product(pid),
7   FOREIGN KEY (bid) REFERENCES buyer(bid)
8 );

```

Results Explain Describe Saved SQL History

	SHID	PID	BID	QUANTITY
2	2	2	2	4
5	5	5	5	5
8	8	8	8	4
7	7	7	7	3
1	1	1	1	4
3	3	3	3	4

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.1.0-15

8) Create table Address

SQL Commands

apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=8723122519658

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Schema WKSP_SAUBER

Language PL/SQL Rows 20 Clear Command Find Tables Save Run

```

1 create table Address (
2   aid INT primary key,
3   phone_number VARCHAR(50),
4   city VARCHAR(50),
5   street VARCHAR(50),
6   house DECIMAL(2,1),
7   entrance INT,
8   apartment INT,
9   foreign key(phone_number) references users(phone_number)
10 );

```

Results Explain Describe Saved SQL History

	AID	PHONE_NUMBER	CITY	STREET	HOUSE	ENTRANCE	APARTMENT
4	5788091415		Illela	Spohn	1.3	1	15
15	9323166198		Concepción Tutuapa	Bay	2.8	2	47
18	4751747133		Pujehue	Del Mar	1.2	2	24
3	1137177958		Verkhniye Achaluki	Gateway	1.6	2	47
7	5131324134		Segezha	Hayes	3	3	47

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.1.0-15

9) Create table Transaction

SQL Commands

apex.oracle.com/pls/apex/r/apex/sql-workshop/sqlcommandprocessor?session=8723122519658

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands Schema WKSP_SAUBER

Language PL/SQL Rows 20 Clear Command Find Tables Save Run

```

1 create table Transaction (
2   tid INT primary key, shid INT,
3   sid INT, aid INT,
4   crid INT, cid INT,
5   starttime DATE, endtime DATE,
6   status VARCHAR(9),
7   foreign key(shid) REFERENCES shopping_cart(shid),
8   foreign key(sid) REFERENCES seller(sid),
9   foreign key(aid) REFERENCES address(aid),
10  foreign key(crid) REFERENCES card(crid),
11  foreign key(cid) REFERENCES courier(cid)
12 );

```

Results Explain Describe Saved SQL History

	TID	SHID	SID	AID	CRID	CID	STARTTIME	ENDTIME	STATUS
2	2	2	2	2	2	2	01/09/2021	-	Pending
1	1	1	1	1	1	1	04/23/2023	04/23/2023	Approved
3	3	3	3	3	3	3	12/02/2023	-	Pending
4	4	4	4	4	4	4	10/26/2022	11/11/2022	Approved

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 23.1.0-15

EXCEPTIONS

FIRST

1.1

```
CREATE OR REPLACE PROCEDURE insert_product (  
    p_pid IN Product.pid%TYPE,  
    p_sid IN Product.sid%TYPE,  
    p_price IN Product.price%TYPE,  
    p_category IN Product.category%TYPE,  
    p_sizes IN Product.sizes%TYPE  
) As  
    category_error EXCEPTION;  
BEGIN  
    IF LENGTH(p_category) < 5 THEN  
        RAISE category_error;  
    ELSE  
        INSERT INTO Product (pid, sid, price, category, sizes) VALUES (p_pid, p_sid, p_price, p_category, p_sizes);  
    END IF;  
EXCEPTION  
    WHEN category_error THEN  
        DBMS_OUTPUT.PUT_LINE('Error: name of category must be at least 5 characters long.');
```

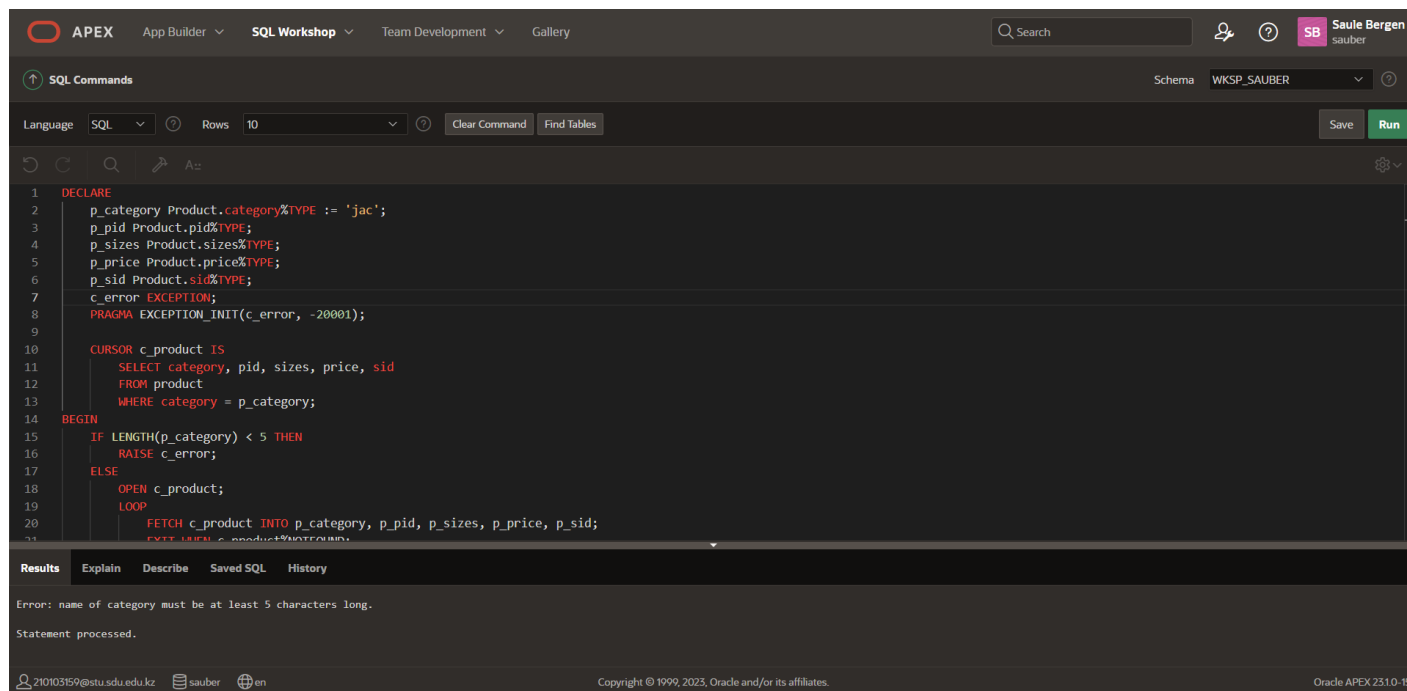
END;

1.2

```
DECLARE  
    p_category Product.category%TYPE := 'Jacket';  
    p_pid Product.pid%TYPE;  
    p_sizes Product.sizes%TYPE;  
    p_price Product.price%TYPE;  
    p_sid Product.sid%TYPE;  
    c_error EXCEPTION;  
    PRAGMA EXCEPTION_INIT(c_error, -20001);  
  
    CURSOR c_product IS
```

```

SELECT category, pid, sizes, price, sid
FROM product
WHERE category = p_category;
BEGIN
IF LENGTH(p_category) < 5 THEN
    RAISE c_error;
ELSE
    OPEN c_product;
    LOOP
        FETCH c_product INTO p_category, p_pid, p_sizes, p_price, p_sid;
        EXIT WHEN c_product%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(p_pid);
        DBMS_OUTPUT.PUT_LINE(p_category);
        DBMS_OUTPUT.PUT_LINE(p_sizes);
        DBMS_OUTPUT.PUT_LINE(p_price);
        DBMS_OUTPUT.PUT_LINE(p_sid);
    END LOOP;
    CLOSE c_product;
END IF;
EXCEPTION
    WHEN c_error THEN
        DBMS_OUTPUT.PUT_LINE('Error: name of category must be at least 5 characters long. ');
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('No such category!');
    WHEN others THEN
        dbms_output.put_line('Error!');
END;
```

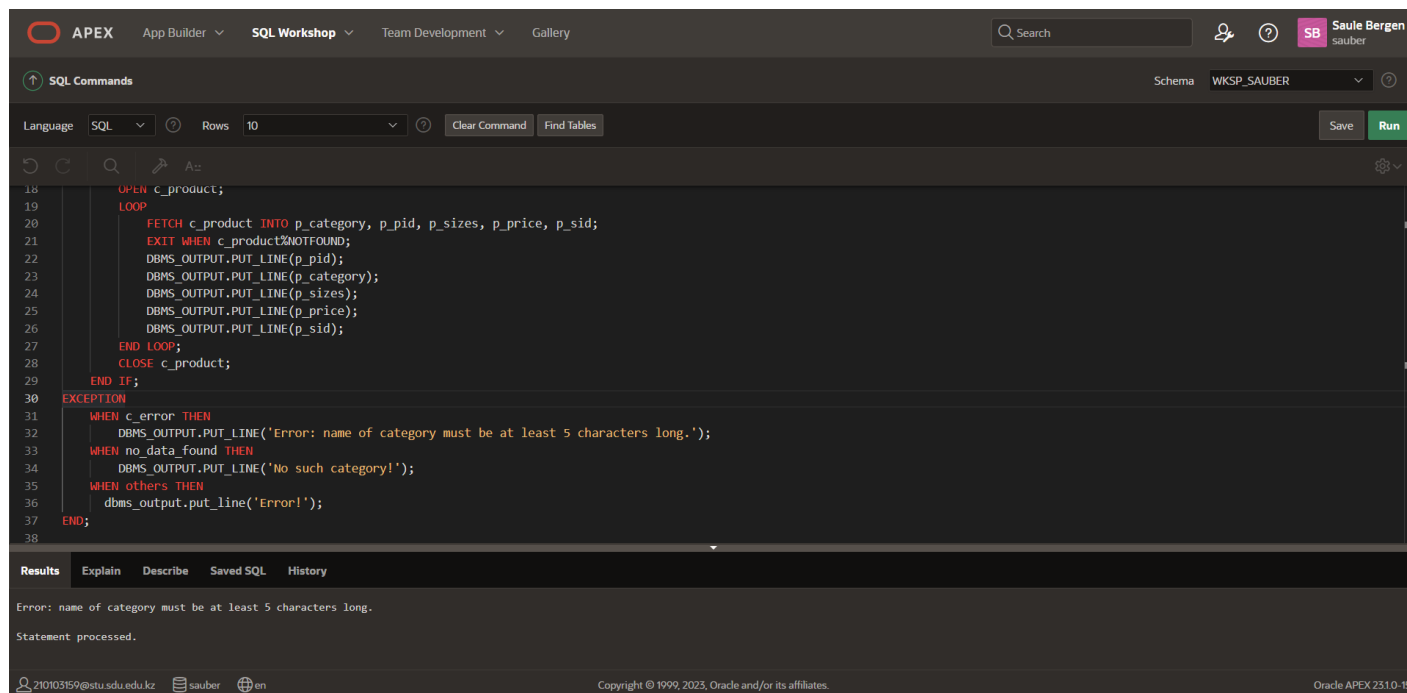


```
1 DECLARE
2   p_category Product.category%TYPE := 'jac';
3   p_pid Product.pid%TYPE;
4   p_sizes Product.sizes%TYPE;
5   p_price Product.price%TYPE;
6   p_sid Product.sid%TYPE;
7   c_error EXCEPTION;
8   PRAGMA EXCEPTION_INIT(c_error, -20001);
9
10  CURSOR c_product IS
11    SELECT category, pid, sizes, price, sid
12    FROM product
13    WHERE category = p_category;
14
15  BEGIN
16    IF LENGTH(p_category) < 5 THEN
17      RAISE c_error;
18    ELSE
19      OPEN c_product;
20      LOOP
21        FETCH c_product INTO p_category, p_pid, p_sizes, p_price, p_sid;
22        EXIT WHEN c_product%NOTFOUND;
```

Results Explain Describe Saved SQL History

Error: name of category must be at least 5 characters long.

Statement processed.



```
18  OPEN c_product;
19  LOOP
20    FETCH c_product INTO p_category, p_pid, p_sizes, p_price, p_sid;
21    EXIT WHEN c_product%NOTFOUND;
22    DBMS_OUTPUT.PUT_LINE(p_pid);
23    DBMS_OUTPUT.PUT_LINE(p_category);
24    DBMS_OUTPUT.PUT_LINE(p_sizes);
25    DBMS_OUTPUT.PUT_LINE(p_price);
26    DBMS_OUTPUT.PUT_LINE(p_sid);
27  END LOOP;
28  CLOSE c_product;
29
30  END IF;
31  EXCEPTION
32    WHEN c_error THEN
33      DBMS_OUTPUT.PUT_LINE('Error: name of category must be at least 5 characters long.');
```

Results Explain Describe Saved SQL History

Error: name of category must be at least 5 characters long.

Statement processed.

SECOND

DECLARE

u_password users.password%TYPE:= 'Abc';

v_let BOOLEAN := FALSE;

v_num BOOLEAN := FALSE;


let EXCEPTION;

num EXCEPTION;

BEGIN

FOR i IN 1..LENGTH(u_password) LOOP

```
IF REGEXP_LIKE(SUBSTR(u_password, i, 1), '[a-zA-Z]') THEN
    v_let := TRUE;
ELSIF REGEXP_LIKE(SUBSTR(u_password, i, 1), '[0-9]') THEN
    v_num := TRUE;
END IF;
END LOOP;
IF v_let AND v_num THEN
    DBMS_OUTPUT.PUT_LINE('Correct');
ELSIF v_let THEN
    raise let;
ELSIF v_num THEN
    raise num;
END IF;
EXCEPTION
    when let then
        DBMS_OUTPUT.PUT_LINE('Add numbers too');
    when num then
        DBMS_OUTPUT.PUT_LINE('Add letters too');
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error! ');
END;
```

 **APEX**

App Builder ▾

SQL Workshop ▾

Team Development ▾

Gallery

↑ SQL Commands

Language SQL ▾ ? Rows 10 ▾ ? Clear Command Find Tables

↶ ↷ 🔍 🛠 A::

```
1 DECLARE
2   u_password users.password%TYPE:= 'Abc';
3   v_let BOOLEAN := FALSE;
4   v_num BOOLEAN := FALSE;
5   let EXCEPTION;
6   num EXCEPTION;
7 BEGIN
8   FOR i IN 1..LENGTH(u_password) LOOP
9     IF REGEXP_LIKE(SUBSTR(u_password, i, 1), '[a-zA-Z]') THEN
10       v_let := TRUE;
11     ELSIF REGEXP_LIKE(SUBSTR(u_password, i, 1), '[0-9]') THEN
12       v_num := TRUE;
13     END IF;
14   END LOOP;
15   IF v_let AND v_num THEN
16     DBMS_OUTPUT.PUT_LINE('Correct');
17   ELSIF v_let THEN
18     raise let;
19   ELSIF v_num THEN
20     raise num;
21   END IF;
22 EXCEPTION
23   when let then
24     DBMS_OUTPUT.PUT_LINE('Add numbers too');
25   when num then
26     DBMS_OUTPUT.PUT_LINE('Add letters too');
27   WHEN OTHERS THEN
28     DBMS_OUTPUT.PUT_LINE('Error!' );
29 END;
30
```

Results

Explain


Describe


Saved SQL


History

Add numbers too

Statement processed.

 210103159@stu.sdu.edu.kz

 sauber

 en

Copyright © 1999, 2023, Oracle and/or its affiliates.

THIRD

DECLARE

```
u_phone_number users.phone_number%type:= '1234567890';
```

```
inval_phone EXCEPTION;
```

BEGIN

```
IF REGEXP_LIKE(u_phone_number, '^\\d{11}$') THEN
```

```
    DBMS_OUTPUT.PUT_LINE('Yoour Phone number is correct: ' || u_phone_number);
```

```
ELSE
```

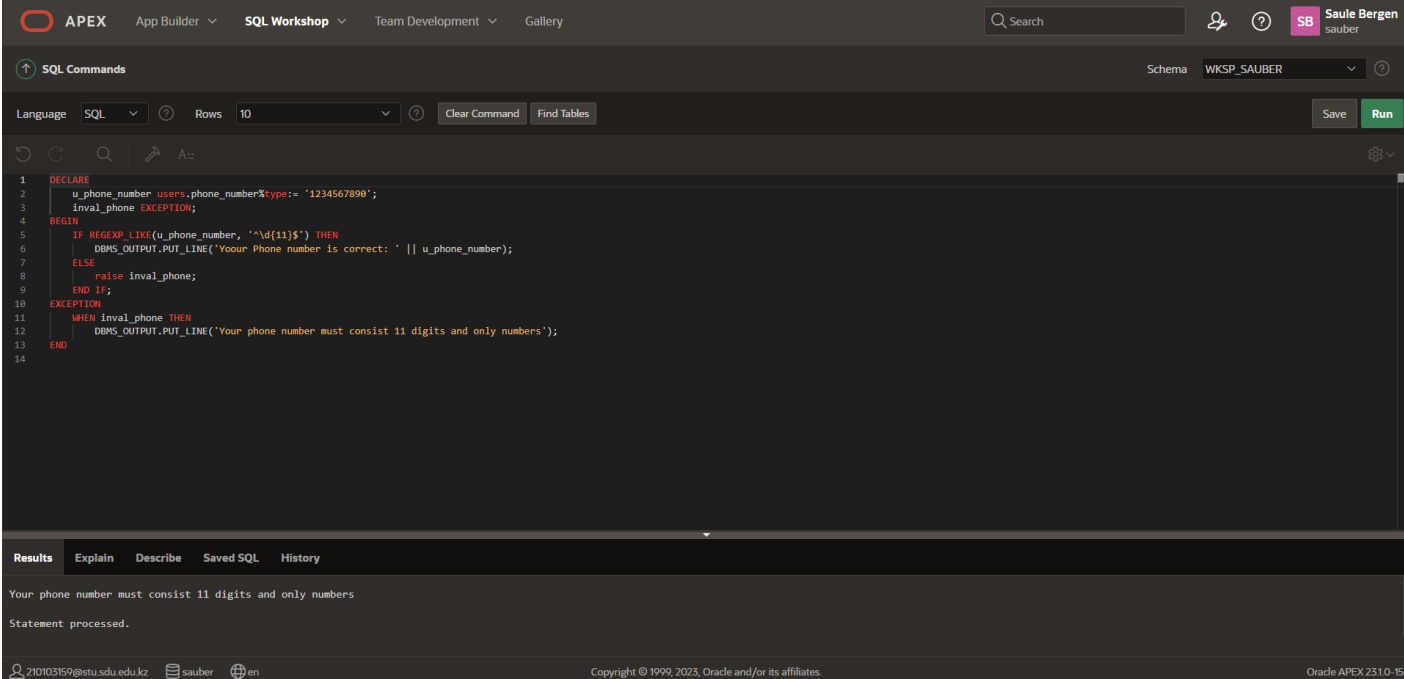
```
    raise inval_phone;
```

```
END IF;
```

EXCEPTION

```
WHEN inval_phone THEN
```

```
DBMS_OUTPUT.PUT_LINE('Your phone number must consist 11 digits and only numbers');  
END;
```



The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'Saula Bergen sauber' are on the right. The 'SQL Commands' tab is active, showing a PL/SQL block with line numbers 1 to 14. The block declares a variable, checks a phone number format, and outputs a message. The 'Run' button is highlighted in green. Below the editor, the 'Results' tab shows the output: 'Your phone number must consist 11 digits and only numbers'. The status bar at the bottom indicates 'Statement processed.' and 'Oracle APEX 2310-15'.

```
1 DECLARE  
2   u_phone_number users.phone_number%type := '1234567890';  
3   inval_phone EXCEPTION;  
4  
5   BEGIN  
6     IF REGEXP_LIKE(u_phone_number, '^d{11}$') THEN  
7       DBMS_OUTPUT.PUT_LINE('Your Phone number is correct: ' || u_phone_number);  
8     ELSE  
9       raise inval_phone;  
10    END IF;  
11  
12    EXCEPTION  
13    WHEN inval_phone THEN  
14      DBMS_OUTPUT.PUT_LINE('Your phone number must consist 11 digits and only numbers');
```

Results Explain Describe Saved SQL History

Your phone number must consist 11 digits and only numbers

Statement processed.

FOURTH

DECLARE

```
u_email Users.email%type := 'john.doe@example.com'; -- replace with your email address  
err EXCEPTION;
```

BEGIN

```
IF INSTR(u_email, '@') = 0 THEN
```

```
    RAISE err;
```

```
ELSE
```

```
    DBMS_OUTPUT.PUT_LINE('Correct email address');
```

```
END IF;
```

EXCEPTION

```
    WHEN err THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Invalid email address');
```

END;

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'Saula Bergen' are on the right. The 'SQL Commands' tab is active, showing a PL/SQL script for email validation. The script declares an exception 'err' and checks if an email address is valid using the INSTR function. If the email is invalid, it raises the exception, which is then caught in the EXCEPTION block to output an error message. The 'Results' tab shows the output 'Correct email address' and 'Statement processed.'.

```
1 DECLARE
2   u_email Users.email%type := 'john.doe@example.com'; -- replace with your email address
3   err EXCEPTION;
4 BEGIN
5   IF INSTR(u_email, '@') = 0 THEN
6     RAISE err;
7   ELSE
8     DBMS_OUTPUT.PUT_LINE('Correct email address');
9   END IF;
10 EXCEPTION
11   WHEN err THEN
12     DBMS_OUTPUT.PUT_LINE('Invalid email address');
13 END;
```

Results: Correct email address. Statement processed.

FIFTH

DECLARE

c_cvv card.cvv%type:= '123';

inval_cvv EXCEPTION;

BEGIN

IF REGEXP_LIKE(c_cvv, '^d{3}\$') THEN

DBMS_OUTPUT.PUT_LINE('Yooour CVV number is correct');

ELSE

raise inval_cvv;

END IF;

EXCEPTION

WHEN inval_cvv THEN

DBMS_OUTPUT.PUT_LINE('Your cvv number must consist 3 digits');

END;


```
1 DECLARE
2     c_cvv card.cvv%type:= '123';
3     inval_cvv EXCEPTION;
4 BEGIN
5     IF REGEXP_LIKE(c_cvv, '^d{3}$') THEN
6         DBMS_OUTPUT.PUT_LINE('Your CVV number is correct');
7     ELSE
8         raise inval_cvv;
9     END IF;
10 EXCEPTION
11     WHEN inval_cvv THEN
12         DBMS_OUTPUT.PUT_LINE('Your cvv number must consist 3 digits');
13 END;
```

Correct email address

Statement processed.

210103159@stu.sdu.edu.kz sauber en Copyright © 1999, 2023, Oracle and/or its affiliates. Oracle APEX 231.0-15

SIXTH

DECLARE

```
c_crID card.crID%type:= '1';
c_phone_number card.phone_number%type;
c_balance card.balance%type;
zero_bal EXCEPTION;
```

BEGIN

```
SELECT balance INTO c_balance
FROM card
WHERE crID = c_crID;
```

```
IF c_balance = 0 THEN
    RAISE zero_bal;
```

ELSE

```
SELECT phone_number, balance INTO c_phone_number, c_balance
FROM card
WHERE crID = c_crID;
```

```

        DBMS_OUTPUT.PUT_LINE(c_phone_number);

        DBMS_OUTPUT.PUT_LINE(c_balance);

    END IF;

EXCEPTION

    WHEN no_data_found THEN

        DBMS_OUTPUT.PUT_LINE('There is no information about this card');

    WHEN zero_bal THEN

        DBMS_OUTPUT.PUT_LINE('Your card is empty');

END;
```

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'Saule Bergen sauber' are on the right. The 'SQL Commands' tab is active, showing a PL/SQL script with line numbers 1 through 26. The script declares variables, selects data from a 'card' table, and includes exception handling for 'no_data_found' and 'zero_bal'. The 'Run' button is highlighted in green. Below the editor, the 'Results' tab shows the output: 'There is no information about this card' and 'Statement processed.' The bottom status bar displays the user '210103159@stu.sdu.edu.kz', the workspace 'sauber', and the Oracle APEX version '2310-15'.

```

1 DECLARE
2   c_crID card.crID%TYPE := '111';
3   c_phone_number card.phone_number%TYPE;
4   c_balance card.balance%TYPE;
5   zero_bal EXCEPTION;
6 BEGIN
7   SELECT balance INTO c_balance
8   FROM card
9   WHERE crID = c_crID;
10
11   IF c_balance = 0 THEN
12     RAISE zero_bal;
13   ELSE
14     SELECT phone_number, balance INTO c_phone_number, c_balance
15     FROM card
16     WHERE crID = c_crID;
17     DBMS_OUTPUT.PUT_LINE(c_phone_number);
18     DBMS_OUTPUT.PUT_LINE(c_balance);
19   END IF;
20 EXCEPTION
21   WHEN no_data_found THEN
22     DBMS_OUTPUT.PUT_LINE('There is no information about this card');
23
24   WHEN zero_bal THEN
25     DBMS_OUTPUT.PUT_LINE('Your card is empty');
26 END;
```

Results | Explain | Describe | Saved SQL | History

There is no information about this card

Statement processed.

210103159@stu.sdu.edu.kz | sauber | en | Copyright © 1999, 2023, Oracle and/or its affiliates. | Oracle APEX 2310-15

SEVENTH

```

CREATE OR REPLACE TRIGGER duplicate_check

BEFORE INSERT OR UPDATE ON address

FOR EACH ROW

DECLARE

    a_AID Address.AID%TYPE;

    duplicate_count INTEGER;
```

```

dupl_error EXCEPTION;
BEGIN
    SELECT COUNT(*) INTO duplicate_count
    FROM address
    WHERE AID = a_AID;

    IF duplicate_count > 0 THEN
        RAISE dupl_error;
    END IF;
EXCEPTION
    when dupl_error THEN
        DBMS_OUTPUT.PUT_LINE('The address of such AID already exists');
END;

```

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

```

CREATE OR REPLACE FUNCTION count_row_f(table_name IN VARCHAR2)
RETURN NUMBER
IS
    count_row NUMBER;
BEGIN
    EXECUTE IMMEDIATE 'SELECT COUNT(*) FROM ' || table_name INTO count_row;
    RETURN count_row;
END count_row_f;

CREATE OR REPLACE TRIGGER count_row_tr
BEFORE INSERT ON users

```

```
FOR EACH ROW
```

```
BEGIN
```

```
DBMS_OUTPUT.PUT_LINE('The current number of rows in the table (not including the row you are adding): ' ||  
count_row_f('users'));
```

```
END;
```

```
CREATE OR REPLACE TRIGGER count_row_tr2
```

```
BEFORE INSERT ON address
```

```
FOR EACH ROW
```

```
BEGIN
```

```
DBMS_OUTPUT.PUT_LINE('The current number of rows in the table (not including the row you are adding): ' ||  
count_row_f('address'));
```

```
END;
```

```
CREATE OR REPLACE TRIGGER count_row_tr3
```

```
BEFORE INSERT ON card
```

```
FOR EACH ROW
```

```
BEGIN
```

```
DBMS_OUTPUT.PUT_LINE('The current number of rows in the table (not including the row you are adding): ' ||  
count_row_f('card'));
```

```
END;
```

```
CREATE OR REPLACE TRIGGER count_row_tr4
```

```
BEFORE INSERT ON seller
```

```
FOR EACH ROW
```

```
BEGIN
```

```
DBMS_OUTPUT.PUT_LINE('The current number of rows in the table (not including the row you are adding): ' ||  
count_row_f('seller'));
```

```
END;
```

```
CREATE OR REPLACE TRIGGER count_row_tr5
```

```
BEFORE INSERT ON buyer
```

```
FOR EACH ROW
```

```
BEGIN
```

```
DBMS_OUTPUT.PUT_LINE('The current number of rows in the table (not including the row you are adding): ' ||  
count_row_f('buyer'));  
  
END;
```

```
CREATE OR REPLACE TRIGGER count_row_tr6  
  
BEFORE INSERT ON shopping_cart  
  
FOR EACH ROW  
  
BEGIN  
  
DBMS_OUTPUT.PUT_LINE('The current number of rows in the table (not including the row you are adding): ' ||  
count_row_f('shopping_cart'));  
  
END;
```

```
CREATE OR REPLACE TRIGGER count_row_tr7  
  
BEFORE INSERT ON transaction  
  
FOR EACH ROW  
  
BEGIN  
  
DBMS_OUTPUT.PUT_LINE('The current number of rows in the table (not including the row you are adding): ' ||  
count_row_f('transaction'));  
  
END;
```

```
CREATE OR REPLACE TRIGGER count_row_tr8  
  
BEFORE INSERT ON courier  
  
FOR EACH ROW  
  
BEGIN  
  
DBMS_OUTPUT.PUT_LINE('The current number of rows in the table (not including the row you are adding): ' ||  
count_row_f('courier'));  
  
END;
```

```
CREATE OR REPLACE TRIGGER count_row_tr9  
  
BEFORE INSERT ON product  
  
FOR EACH ROW  
  
BEGIN  
  
DBMS_OUTPUT.PUT_LINE('The current number of rows in the table (not including the row you are adding): ' ||  
count_row_f('product'));  
  
END;
```

```
1 CREATE OR REPLACE FUNCTION count_row_f(table_name IN VARCHAR2)
2 RETURN NUMBER
3 IS
4     count_row NUMBER;
5 BEGIN
17 CREATE OR REPLACE TRIGGER count_row_tr2
18 BEFORE INSERT ON address
19 FOR EACH ROW
84 Insert into users(PHONE_NUMBER,NAME, SURNAME, EMAIL, PASSWORD, REGISTRATIONDATE, STATUS)
85 values ('9803372670', 'Mika', 'Joly' , 'meru_j@mail.com' , 'mika_j12345', '1/30/2021', 'buyer');
86
87 SELECT count(*) FROM users;
```

Results Explain Describe Saved SQL History

The current number of rows in the table (not including the row you are adding): 24

1 row(s) inserted.

Results Explain Describe Saved SQL History

Trigger created.

0.04 seconds

```
87 SELECT count(*) FROM users;
```

Results Explain Describe Saved SQL History

25

Create trigger status upgrade

Trigger change status where we update end or start time in table transaction.

Approved: if start time is before 14 hours from end time Declined: if start time is after 14 hours from end time Pending: if start and end time are within 14 hours

SQL Commands

Language PL/SQL Rows 20 Clear Command Find Tables

↶ ↷ 🔍 ↵ A:

```
1 CREATE OR REPLACE TRIGGER status_upgrade
2 BEFORE UPDATE ON transaction
3 FOR EACH ROW
4 DECLARE
5     date_dif NUMBER;
6 BEGIN
7     date_dif := 14;
8     IF :new.endtime - :new.startTime >= date_dif THEN
9         :new.status := 'Declined';
10    ELSIF :new.endtime - :new.startTime < date_dif THEN
11        :new.status := 'Approved';
12    ELSE
13        :new.status := 'Pending';
14    END IF;
15 END;
```

Results Explain Describe Saved SQL History

Trigger created.

0.12 seconds

210105159@stu.sdu.edu.kz sauber en

Copyright © 1999, 2023, Oracle and/or its affiliates.

APEX App Builder SQL Workshop Team Development Gallery Search

SQL Commands Schema WKSP_SAUBER

Language PL/SQL Rows 20 Clear Command Find Tables Save Run

↶ ↷ 🔍 ↵ A:

```
1 select * from transaction where tid = 1;
2
```

Results Explain Describe Saved SQL History

TID	SHID	SID	AID	CRID	CID	STARTTIME	ENDTIME	STATUS
1	1	1	1	1	1	04/23/2023	04/23/2023	Approved

1 rows returned in 0.00 seconds Download

APEX App Builder SQL Workshop Team Development Gallery Search

SQL Commands Schema WKSP_SAUBER

Language PL/SQL Rows 20 Clear Command Find Tables Save Run

↶ ↷ 🔍 ↵ A:

```
1 update transaction set endTime = '05/15/2023' where tid = 2;
2
```

Results Explain Describe Saved SQL History

1 row(s) updated.

0.02 seconds

APEX App Builder SQL Workshop Team Development Gallery Search

Schema WKSP_SAUBER

Language PL/SQL Rows 20 Clear Command Find Tables Save Run

```

1 select * from transaction where tid = 2;
2

```

Results Explain Describe Saved SQL History

TID	SHID	SID	AID	CRID	CID	STARTTIME	ENDTIME	STATUS
2	2	2	2	2	2	01/09/2021	05/15/2023	Declined

1 rows returned in 0.00 seconds Download

Create procedure add new user

We add new user to our db. And table add this user by their status.

If add courier status, add in table courier too.

APEX App Builder SQL Workshop Team Development Gallery Search

Schema WKSP_SAUBER

Language PL/SQL Rows 20 Clear Command Find Tables

```

1 CREATE OR REPLACE PROCEDURE addUser(
2   new_phone_number users.phone_number%TYPE,
3   new_name users.name%TYPE,
4   new_surname users.surname%TYPE,
5   new_email users.email%TYPE,
6   new_password users.password%TYPE,
7   new_date users.registrationDate%TYPE,
8   new_status users.status%TYPE
9 ) AS
10  new_bid buyer.bid%TYPE;
11  new_sid seller.sid%TYPE;
12  new_cid courier.cid%TYPE;
13 BEGIN
14  INSERT INTO users (phone_number, name, surname, email, password, registrationDate, status)
15  VALUES (new_phone_number, new_name, new_surname, new_email, new_password, new_date, new_status);
16  IF new_status = 'buyer' THEN
17    SELECT MAX(bid) + 1 INTO new_bid FROM buyer;
18    INSERT INTO buyer (bid, phone_number) VALUES (new_bid, new_phone_number);
19  ELSIF new_status = 'seller' THEN
20    SELECT MAX(sid) + 1 INTO new_sid FROM seller;
21    INSERT INTO seller (sid, rating) VALUES (new_sid, 0);
22  ELSIF new_status = 'courier' THEN
23    SELECT MAX(cid) + 1 INTO new_cid FROM courier;
24    INSERT INTO courier (cid, rating) VALUES (new_cid, 0);
25  ELSE

```

Results Explain Describe Saved SQL History

Procedure created.

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Copyright © 1999, 2023, Oracle and/or its affiliates.

APEX App Builder SQL Workshop Team Development Gallery Search

Schema WKSP_SAUBER

Language PL/SQL Rows 20 Clear Command Find Tables Save Run

```

1 BEGIN
2   addUser('1803372590', 'John', 'Doe', 'johndoe@example.com', 'password', '04/22/2021', 'buyer');
3 END;
4 /

```

Results Explain Describe Saved SQL History

The current number of rows in the table (not including the row you are adding): 26
The current number of rows in the table (not including the row you are adding): 23

Statement processed.

0.00 seconds

TRANSACTION

In particular, it selects a courier with a rating of 4 or higher that has not delivered any transactions in the last 14 days, selects the price of an item with a specific ID, updates the status and end time of a transaction with a specific ID to "Approved" and the current date, and updates card balance with a specific identifier associated with the selected courier. If the courier is unavailable or an error occurs, then throws an application error or rolls back the changes and throws an error again.

```
1 DECLARE
2   r_tid Transaction.tid%TYPE := 1;
3   r_crid Card.crid%TYPE;
4   r_price product.price%TYPE;
5 BEGIN
6   SELECT cid INTO r_crid
7   FROM courier
8   WHERE rating >= 4.0
9   AND cid NOT IN (
10    SELECT cid
11    FROM Transaction
12    WHERE status = 'Courier'
13    AND endTime BETWEEN SYSDATE - 14 AND SYSDATE
14  )
15   AND ROWNUM = 1;
16   SELECT price INTO r_price
17   FROM product
18   WHERE pid = r_tid;
19
20   UPDATE Transaction
21   SET status = 'Approved',
22       endTime = SYSDATE
```

Results: Explain Describe Saved SQL History

1 row(s) updated.

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Transaction 2.

```
2   SELECT crid, balance INTO v_crid, v_balance
3   FROM Card WHERE phone_number = (
4     SELECT phone_number FROM Users WHERE phone_number = (
5       SELECT phone_number FROM Buyer WHERE bid = v_bid));
6
7   SELECT price INTO v_price FROM Product
8   WHERE pid = v_pid;
9   v_total_price := v_price * v_quantity;
10  BEGIN
11    UPDATE Card SET balance = v_balance - v_total_price
12    WHERE crid = v_crid;
13    UPDATE Card
14    SET balance = balance + v_total_price WHERE phone_number = (
15      SELECT phone_number FROM Users WHERE phone_number = (
16        SELECT phone_number FROM Seller WHERE sid = v_sid));
17
18    UPDATE Transaction SET status = 'Completed', endTime = SYSDATE
19    WHERE shid = v_shid;
20    INSERT INTO Log (phone_number, date_, info)
21    VALUES (( SELECT phone_number FROM Users
22      WHERE phone_number = ( SELECT phone_number FROM Buyer
23        WHERE bid = v_bid)), SYSDATE, 'transaction');
24    COMMIT;
25
26    DBMS_OUTPUT.PUT_LINE('Products have been successfully purchased: ' || v_quantity || ' product unit with ID ' || v_pid || 'for the amount of ' || v_total_price || ' tenge.');
```

EXCEPTION

```
8   WHEN OTHERS THEN
9     ROLLBACK;
10    DBMS_OUTPUT.PUT_LINE('Transaction execution error: ' || SQLERRM); END;
```

1 END;

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:

```
190
191  --return average price of product category
192  create or replace procedure my_procedure is
193  cursor my_cursor is
194  select category, AVG(price) as avg_price
195  from product
196  group by category;
197  v_category product.category%TYPE;
198  v_avg_price product.price%TYPE;
199  begin
200  open my_cursor;
201  loop
202  fetch my_cursor into v_category, v_avg_price;
203  exit when my_cursor%NOTFOUND;
204  dbms_output.put_line(v_category || ': ' || v_avg_price);
205  end loop;
206  close my_cursor;
207  end;
208
209  begin
210  my_procedure;
211  end;
```

And this is the result of our function:

```
198  v_avg_price prod
Results Explain Desc
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:

This is the code:

```
115
116  --notifies you of a successful procedure
117  create or replace procedure update_price(
118      pr_id in number,
119      new_price in number
120  )
121  is
122  begin
123      update product set price = new_price
124      where pid = pr_id;
125      if SQL%ROWCOUNT = 0 then
126          RAISE_APPLICATION_ERROR(-20002, 'Product not found');
127      else
128          dbms_output.put_line('Price updated successfully');
129      end if;
130  exception
131  when others then
132      dbms_output.put_line('Error: ' || SQLERRM);
133  end;
134
135  begin
136      update_price(10, 26990);
137  end;
```

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

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

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

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

```

92  --if we enter product id that procedure return us other information about product
93  create or replace procedure get_product(v_pid product.pid%TYPE)
94  as
95  v_size product.sizes%TYPE;
96  v_category product.category%TYPE;
97  v_price product.price%TYPE;
98  begin
99  select sizes, category, price into v_size, v_category, v_price
100  from product
101  where pid = v_pid;
102  dbms_output.put_line('Product information');
103  dbms_output.put_line('size: ' || v_size);
104  dbms_output.put_line('category: ' || v_category);
105  dbms_output.put_line('price: ' || v_price);
106  exception
107  when NO_DATA_FOUND then
108  dbms_output.put_line('Record not found');
109  end;
110
111  begin
112  get_product(546);
113  end;
114

```

This is the result of procedure:

Results	Explain	Desc
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(*)):

```

61
62  --declare how many users our shop have
63  create or replace function count_users
64  return number
65  is
66  v_users_count number;
67  begin
68  select count(*) into v_users_count
69  from users;
70  return v_users_count;
71  end;
72
73  declare
74  res number;
75  begin
76  res:=count_users;
77  dbms_output.put_line('Count of users is ' || res);
78  end;
79

```

This is the result of function:

Results	Explain	Describe
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:

```

167  --return us average price of current category
168  create or replace function avg_price(category varchar2) return number
169  is
170  total number := 0;
171  product_count number := 0;
172  begin
173  for prod in (select price from product where category = category)
174  loop total := total + prod.price;
175  product_count := product_count + 1;
176  end loop;
177  if product_count = 0 then
178  return 0;
179  else
180  return total / product_count;
181  end if;
182  end;
183
184  declare
185  res number;
186  begin
187  res := avg_price('Skirt');
188  dbms_output.put_line('Average price of skirts is: ' || res);
189  end;

```

Result of this function is:

Results	Explain	Describe	Saved SQL
Average price of skirts is: 20708.25			
Statement processed.			
0.02 seconds			

