ENDTERM

DBMS

Introduction

Users

Seller

Buyer

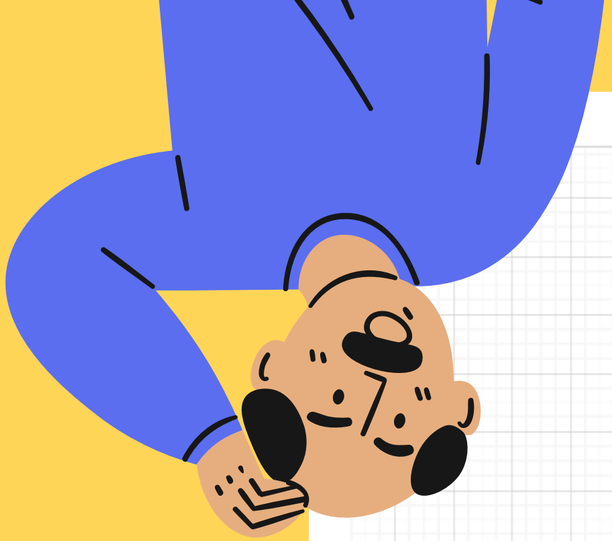
Orders

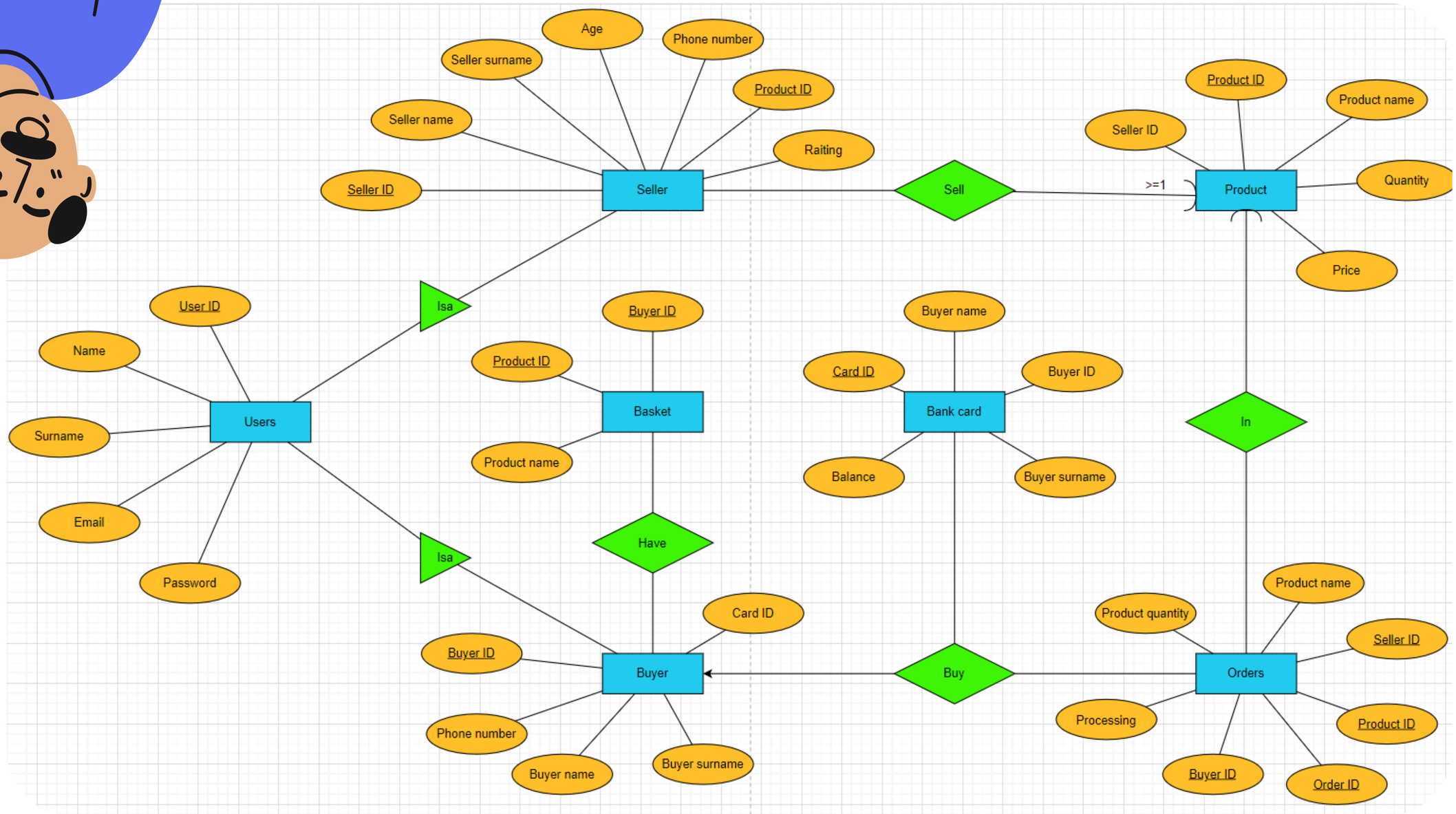
Products

Basket

Bank card

FIRST, I CREATED 7 TABLES, THAT IS, 7 ENTITIES IN ERD, WITH THEIR ATTRIBUTES.




Age
Phone number
Seller surname
Product ID
Seller name
Raiting
Seller ID
Seller
Sell
>=1
Product ID
Product name
Seller ID
Product
Quantity
Price
Isa
User ID
Buyer ID
Name
Product ID
Basket
Users
Surname
Product name
Email
Password
Have
Isa
Buyer name
Card ID
Buyer ID
Bank card
Balance
Buyer surname
In
Card ID
Product name
Product quantity
Buyer ID
Buyer
Buy
Orders
Seller ID
Phone number
Processing
Product ID
Buyer name
Buyer surname
Buyer ID
Order ID

User_id	Name	Surname	Email	Password	Type
1	Cullan	Devote	cdevote0@weibo.com	KikuXE	Seller
2	Vasili	Hubner	vhubner1@reddit.com	dPWVGP	Buyer
3	Caro	Alenichicov	calenichicov2@sina.com.cn	MfpSswbs	Buyer
4	Claire	Roseby	croseby3@skype.com	zN8ve0QX	Seller
5	Pierrette	Frugier	pfrugier4@wired.com	BuZ9BWqdP	Buyer
6	Lara	Youell	lyouell5@dailymail.co.uk	zuu4aCKrVxXn	Seller
7	Kamilah	Longland	klongland6@fema.gov	b9c28uyGA	Buyer
8	Christophorus	Russon	crusson7@ed.gov	utPsqBIc8o5	Seller
9	Byram	Hatherley	bhatherley8@weebly.com	UNimiMoB	Seller
10	Laure	Roskeilly	lroskeilly9@t-online.de	GGJYHg	Buyer
11	Bennett	Blakden	bblakdena@independent.co.uk	1Zi20ov0Z	Buyer
12	Obadias	Yurkiewicz	oyurkiewiczb@ebay.co.uk	RlwdYfEe	Seller
13	Cheslie	Christie	cchristiec@elegantthemes.com	U4NGuC	Seller
14	Margery	Behling	mbehlingd@washingtonpost.com	6GbL4i	Seller
15	Cazzie	Lamminam	clamminame@oracle.com	tiw7RggG	Seller
16	Gustaf	Meere	gmeeref@engadget.com	m8itY3T	Buyer
17	Sherry	Mapis	smapisg@xinhuanet.com	QiutfzDL2	Buyer

First normal form

- **The table must not contain duplicate rows**
- **No arrays or lists of any kind**
- **A column contains data of the same type**

ɔser_id	Name	Surname	Email	Password	Type
1	Cullan	Devote	cdevote0@weibo.com	KikuXE	Seller
2	Vasili	Hubner	vhubner1@reddit.com	dPWVGP	Buyer
3	Caro	Alenichicov	calenichicov2@sina.com.cn	MfpSswbs	Buyer
4	Claire	Roseby	croseby3@skype.com	zN8ve0QX	Seller
5	Pierrette	Frugier	pfrugier4@wired.com	BuZ9BWqdP	Buyer
6	Lara	Youell	lyouell5@dailymail.co.uk	zuu4aCKrVxXn	Seller
7	Kamilah	Longland	klongland6@fema.gov	b9c28uyGA	Buyer
8	Christophorus	Russon	crusson7@ed.gov	utPsqBIc8o5	Seller
9	Byram	Hatherley	bhatherley8@weebly.com	UNimiMoB	Seller
10	Laure	Roskeilly	lroskeilly9@t-online.de	GGJYHg	Buyer
11	Bennett	Blakden	bblakdena@independent.co.uk	1Zi20ov0Z	Buyer
12	Obadias	Yurkiewicz	oyurkiewiczb@ebay.co.uk	RlwdYfEe	Seller
13	Cheslie	Christie	cchristiec@elegantthemes.com	U4NGuC	Seller
14	Margery	Behling	mbehlingd@washingtonpost.com	6GbL4i	Seller
15	Cazzie	Lamminam	clamminame@oracle.com	tiw7RggG	Seller
16	Gustaf	Meere	gmeeref@engadget.com	m8itY3T	Buyer
17	Sherry	Mapis	smapisg@xinhuanet.com	QiutfzDL2	Buyer

Second normal form

- **The table must be in first normal form**
- **The table must have a primary key**
- **All non-key columns of the table depend on the primary key**

ɔser_id	Name	Surname	Email	Password	Type
1	Cullan	Devote	cdevote0@weibo.com	KikuXE	Seller
2	Vasili	Hubner	vhubner1@reddit.com	dPWVGP	Buyer
3	Caro	Alenichicov	calenichicov2@sina.com.cn	MfpSswbs	Buyer
4	Claire	Roseby	croseby3@skype.com	zN8ve0QX	Seller
5	Pierrette	Frugier	pfrugier4@wired.com	BuZ9BWqdP	Buyer
6	Lara	Youell	lyouell5@dailymail.co.uk	zuu4aCKrVxXn	Seller
7	Kamilah	Longland	klongland6@fema.gov	b9c28uyGA	Buyer
8	Christophorus	Russon	crusson7@ed.gov	utPsqBIc8o5	Seller
9	Byram	Hatherley	bhatherley8@weebly.com	UNimiMoB	Seller
10	Laure	Roskeilly	lroskeilly9@t-online.de	GGJYHg	Buyer
11	Bennett	Blakden	bblakdena@independent.co.uk	1Zi20ov0Z	Buyer
12	Obadias	Yurkiewicz	oyurkiewiczb@ebay.co.uk	RlwdYfEe	Seller
13	Cheslie	Christie	cchristiec@elegantthemes.com	U4NGuC	Seller
14	Margery	Behling	mbehlingd@washingtonpost.com	6GbL4i	Seller
15	Cazzie	Lamminam	clamminame@oracle.com	tiw7RggG	Seller
16	Gustaf	Meere	gmeeref@engadget.com	m8itY3T	Buyer
17	Sherry	Mapis	smapisg@xinhuanet.com	QiutfzDL2	Buyer

Third normal form

The requirement of the third normal form is that there is no transitive dependence in the tables.

Transitive dependency is when non-key columns depend on the values of other non-key columns. That is, non-key columns must depend only on the primary key

ɔser_id	Name	Surname	Email	Password	Type
1	Cullan	Devote	cdevote0@weibo.com	KikuXE	Seller
2	Vasili	Hubner	vhubner1@reddit.com	dPWVGP	Buyer
3	Caro	Alenichicov	calenichicov2@sina.com.cn	MfpSswbs	Buyer
4	Claire	Roseby	croseby3@skype.com	zN8ve0QX	Seller
5	Pierrette	Frugier	pfrugier4@wired.com	BuZ9BWqdP	Buyer
6	Lara	Youell	lyouell5@dailymail.co.uk	zuu4aCKrVxXn	Seller
7	Kamilah	Longland	klongland6@fema.gov	b9c28uyGA	Buyer
8	Christophorus	Russon	crusson7@ed.gov	utPsqBIc8o5	Seller
9	Byram	Hatherley	bhatherley8@weebly.com	UNimiMoB	Seller
10	Laure	Roskeilly	lroskeilly9@t-online.de	GGJYHg	Buyer
11	Bennett	Blakden	bblakdena@independent.co.uk	1Zi20ov0Z	Buyer
12	Obadias	Yurkiewicz	oyurkiewiczb@ebay.co.uk	RlwdYfEe	Seller
13	Cheslie	Christie	cchristiec@elegantthemes.com	U4NGuC	Seller
14	Margery	Behling	mbehlingd@washingtonpost.com	6GbL4i	Seller
15	Cazzie	Lamminam	clamminame@oracle.com	tiw7RggG	Seller
16	Gustaf	Meere	gmeeref@engadget.com	m8itY3T	Buyer
17	Sherry	Mapis	smapisg@xinhuanet.com	QiutfzDL2	Buyer

Users

User_id	Name	Surname	Email	Password	Type
1	Cullan	Devote	cdevote0@weibo.com	KikuXE	Seller
2	Vasili	Hubner	vhubner1@reddit.com	dPWVGP	Buyer
3	Caro	Alenichicov	calenichicov2@sina.com.cn	MfpSswbs	Buyer
4	Claire	Roseby	croseby3@skype.com	zN8ve0QX	Seller
5	Pierrette	Frugier	pfrugier4@wired.com	BuZ9BWqdP	Buyer
6	Lara	Youell	lyouell5@dailymail.co.uk	zuu4aCKrVxXn	Seller
7	Kamilah	Longland	klongland6@fema.gov	b9c28uyGA	Buyer
8	Christophorus	Russon	crusson7@ed.gov	utPsqBIc8o5	Seller
9	Byram	Hatherley	bhatherley8@weebly.com	UNimiMoB	Seller
10	Laure	Roskeilly	lroskeilly9@t-online.de	GGJYHg	Buyer
11	Bennett	Blakden	bblakdena@independent.co.uk	1Zi20ov0Z	Buyer
12	Obadias	Yurkiewicz	oyurkiewiczb@ebay.co.uk	RlwdYfEe	Seller
13	Cheslie	Christie	cchristiec@elegantthemes.com	U4NGuC	Seller
14	Margery	Behling	mbehlingd@washingtonpost.com	6GbL4i	Seller
15	Cazzie	Lamminam	clamminame@oracle.com	tiw7RggG	Seller
16	Gustaf	Meere	gmeeref@engadget.com	m8itY3T	Buyer
17	Sherry	Mapis	smapisg@xinhuanet.com	QiutfzDL2	Buyer

Seller

A	B	C	D	E	F	G
Seller_ID	Name	Surname	Email	Age	**Phone number**	Rating
1	Cullan	Devote	cdevote0@weibo.com	24	8402653880	74
4	Claire	Roseby	croseby3@skype.com	32	2055611989	97
6	Lara	Youell	lyouell5@dailymail.co.uk	45	3919081882	85
8	Christophorus	Russon	crusson7@ed.gov	21	2531098218	65
9	Byram	Hatherley	bhatherley8@weebly.com	19	1607126591	96
12	Obadias	Yurkiewicz	oyurkiewiczb@ebay.co.uk	25	5274826055	82
13	Cheslie	Christie	cchristiec@elegantthemes.com	36	6547165180	71
14	Margery	Behling	mbehlingd@washingtonpost.com	45	5858818913	79
15	Cazzie	Lamminam	clamminame@oracle.com	30	5577018413	63
21	Arni	Dietzler	adietzlerk@dedecms.com	25	9074029208	95
25	Rivkah	Kirby	rkirbyo@list-manage.com	24	7645941937	74
26	Eleni	Sperwell	esperwellp@census.gov	29	9121043876	65
27	Judas	Lodewick	jlodewickq@cam.ac.uk	27	3613921817	89
31	Minerva	Kowal	mkowalu@linkedin.com	54	5646836403	65
32	Bastien	Cessford	bcessfordv@state.tx.us	30	1394044320	75
36	Arabelle	Hawkswood	ahawkswoodz@hexun.com	21	2199778720	78
38	Any	Dyster	adyster11@cloudflare.com	25	7223139033	95
39	Zak	Crannage	zcrannage12@newyorker.com	24	5405980886	62
40	Heida	Gouldstone	hgouldstone13@comcast.net	23	3045062880	95
46	Winn	O'Heagertie	woheagertie19@toplist.cz	26	6263037823	71

Buyer

Buyer_ID	Name	Surname	**Phone number**	Card_ID
2	Vasili	Hubner	7737424170	45279529
3	Caro	Alenichicov	5218241464	49762064
5	Pierrette	Frugier	4243734908	46413279
7	Kamilah	Longland	6869036971	49472402
10	Laure	Roskeilly	1299604191	49630147
11	Bennett	Blakden	5205345813	47304997
16	Gustaf	Meere	9127180279	43725952
17	Sherry	Mapis	6882307867	47496574
18	Rudiger	Carslake	4769697920	41601314
19	Annabel	Ioselev	6173020048	46995810
20	Frederick	Senogles	8072656179	45044489
22	Domenico	Grogan	6036840077	41343361
23	Marketa	Guiot	7308016149	41597350

Orders

Order_ID	Buyer_ID	Seller_ID	**Product_ID**	Product_name	Product_Quantity	Processing
993566	2	1	50	Shirts	4	delivered
760364	3	4	51	Jeans	7	sent
421608	5	6	52	Blouses	6	delivered
540928	7	8	53	T-shirts	4	sent
670624	10	9	54	Dresses	8	delivered
597181	11	12	55	Pants	4	delivered
539338	16	13	56	Shorts	6	delivered
230631	17	14	57	Sweatshirts	7	sent
571925	18	15	58	Underwear	8	delivered
598041	19	21	59	Swimwear	10	process
704855	20	25	60	Swimsuits	4	sent
165312	22	26	61	Jackets	5	sent
351756	23	27	62	Suit	9	sent
948564	24	31	63	Overalls	9	process
621993	28	32	64	Corsets	8	delivered
842891	29	36	65	Skirts	2	sent

Products

Product_ID	Seller_ID	Product_name	Quantity	Price
50	1	Shirts	120	22750
51	4	Jeans	134	48490
52	6	Blouses	90	19730
53	8	T-shirts	85	54760
54	9	Dresses	95	12020
55	12	Pants	94	48520
56	13	Shorts	133	17750
57	14	Sweatshirts	61	20190
58	15	Underwear	117	11090
59	21	Swimwear	128	21770
60	25	Swimsuits	111	32600

Basket

Buyer_ID	Product_ID	Product_name
2	50	Shirts
3	51	Jeans
5	52	Blouses
7	53	T-shirts
10	54	Dresses
11	55	Pants

Bank card

Card_ID	Buyer_ID	Name	Surname	Balance
45279529	2	Vasili	Hubner	854100
49762064	3	Caro	Alenichicov	848200
46413279	5	Pierrette	Frugier	337100
49472402	7	Kamilah	Longland	956500
49630147	10	Laure	Roskeilly	548400
47304997	11	Bennett	Blakden	515600
43725952	16	Gustaf	Meere	993900
47496574	17	Sherry	Mapis	428300
41601314	18	Rudiger	Carslake	895600
46995810	19	Annabel	Ioselev	928600
45044489	20	Frederick	Senogles	643900
41343361	22	Domenico	Grogan	10510[illegible]

Procedure which does group by information

```
CREATE OR REPLACE PROCEDURE how_much (
tip varchar,
total OUT NUMBER
) AS
BEGIN
    Select count(USER_ID) into total from USERS
        WHERE USERS.TYPE = tip group by USERS.TYPE;
        DBMS_OUTPUT.PUT_LINE( A: total || ' ' || tip || 's');
END;

DECLARE
    who varchar2(255) := 'Buyer';
    total number;
BEGIN
    how_much( tip: who,  total: total);
end;
```

Function which counts the number of records

```
create or replace function numbers(
col varchar
)
return number as
    total number;
Begin
    select count(col) into total from BUYER;
    return total;
end;

DECLARE
    col varchar2(255) := 'NAME';
    res number;
Begin
    res := numbers( col: col);
    DBMS_OUTPUT.PUT_LINE( A: 'The number of records: ' || res);
end;
```

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

```
CREATE OR REPLACE PROCEDURE insert_buyer (
    b_id number,
    b_name varchar,
    b_surname varchar,
    b_phone number,
    b_card_id number,
    res out number
) AS
BEGIN
INSERT INTO BUYER VALUES (
b_id, b_name, b_surname, b_phone, b_card_id
);
res := sql%rowcount;
END;

DECLARE
    res number;
    resall number := 0;
BEGIN
    insert_buyer( b_id: 51, b_name: 'Assanali', b_surname: 'Rakhimov', b_phone: 88005553535, b_card_id: 444555666, res: res);
    resall := resall + res;
    DBMS_OUTPUT.PUT_LINE( A: resall);
end;
```

Add user-defined exception which disallows to enter title of item to be less than 3 characters

```
CREATE OR REPLACE TRIGGER short_name
BEFORE INSERT ON products
FOR EACH ROW
BEGIN
    IF LENGTH(:NEW.PRODUCT_NAME) < 3 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Length of name should be longer than 3 symbol');
    END IF;
END;
```

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

```
CREATE OR REPLACE TRIGGER insert_to_users
BEFORE INSERT ON USERS
FOR EACH ROW
DECLARE
    counting number;
BEGIN
    select count(USER_ID) into counting from USERS;
    DBMS_OUTPUT.PUT_LINE( A: 'Number of row that already have: ' || counting);
END;
```