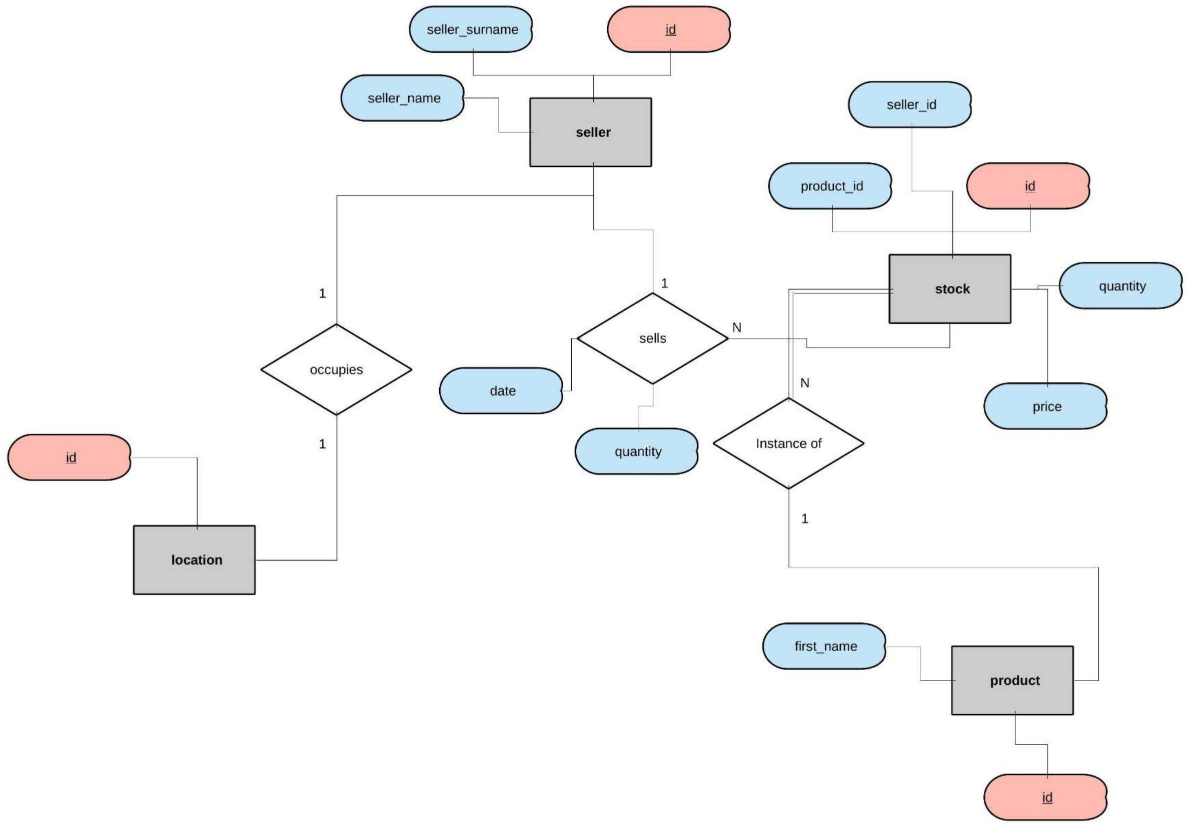
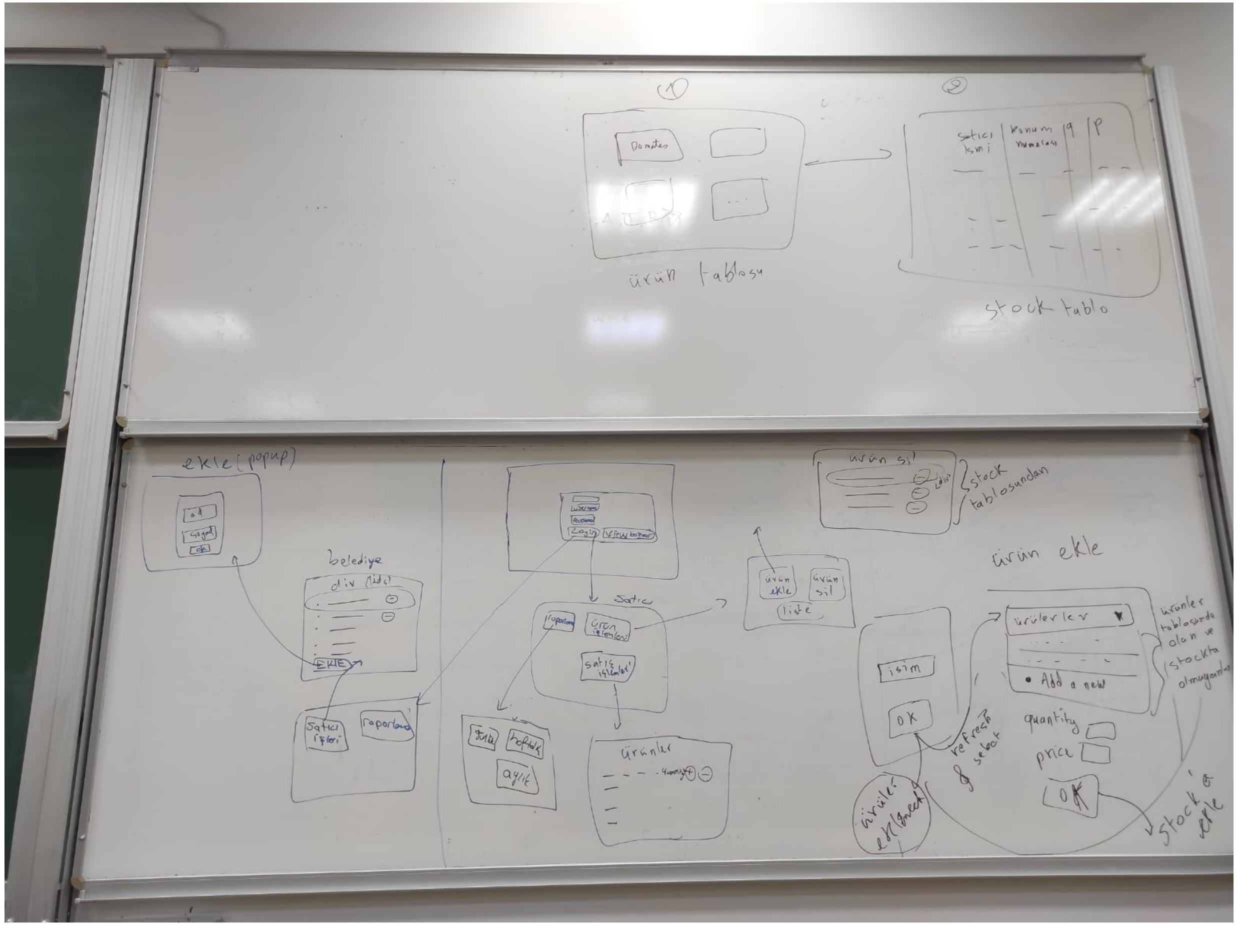
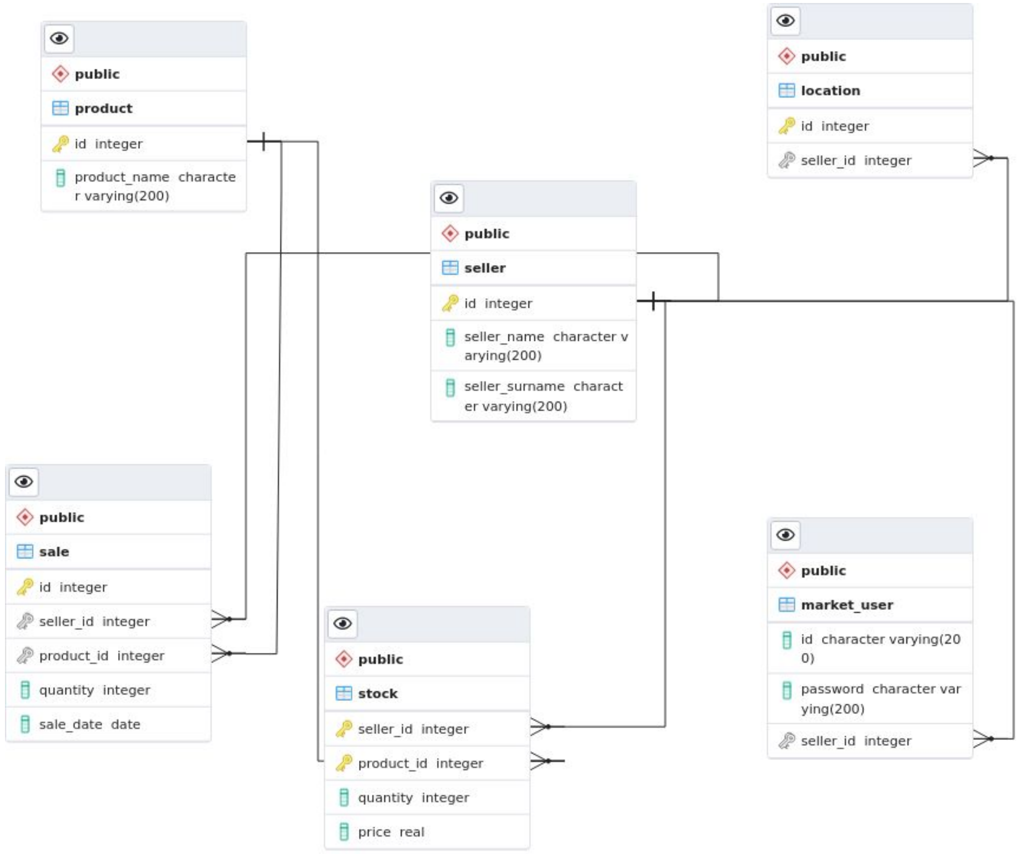
ER Diagram


seller_surname
id
seller_name
seller
seller_id
product_id
id
stock
quantity
price
1
1
sells
N
occupies
date
quantity
N
Instance of
1
1
id
location
1
first_name
product
id

Design Stage



Schema



public
product
id integer
product_name character varying(200)
public
location
id integer
seller_id integer
public
seller
id integer
seller_name character varying(200)
seller_surname character varying(200)
public
sale
id integer
seller_id integer
product_id integer
quantity integer
sale_date date
public
stock
seller_id integer
product_id integer
quantity integer
price real
public
market_user
id character varying(200)
password character varying(200)
seller_id integer

İstenen Maddeler

Tablolarınızda primary key ve foreign key kısıtlarını kullanmalısınız.

```sql
create table if not exists sale (
    id int primary key,
    seller_id int not null references seller (id) on delete cascade,
    product_id int not null references product (id) on delete cascade,
    quantity int not null,
    sale_date date not null
);
```

En az bir tabloda silme kısıtı ve sayı kısıtı olmalıdır.

```
create table if not exists stock (
    seller_id int references seller (id) on delete cascade,
    product_id int references product (id) on delete cascade,
    quantity int not null,
    price real not null,
    constraint pk_stock primary key (product_id, seller_id),
    check (price > 0),
    check (quantity > - 1)
);
```

Arayüzden en az birer tane insert, update ve delete işlemi gerçekleştirilmeli.

```
create or replace function add_product (name varchar(200), out productId int)
as $$
declare
    v_msg text;
    next_pid int;
begin
    next_pid := nextval('productid_seq');
    insert into product (id, product_name)
        values (next_pid, name);
    productId := next_pid;
end;
$$
language 'plpgsql';
```

Arayüzden girilecek bir değere göre ekrana sonuçların listelendiği bir sorgu.

```
create or replace function list_stock (sellerId int)
   returns table (productid int, productname varchar(200), quantity int, price real)
   as $$
begin
   return query
   select
      stock.product_id,
      product.product_name,
      stock.quantity,
      stock.price
   from stock, product
   where stock.seller_id = sellerId and product.id = stock.product_id;
end;
$$
language 'plpgsql';
```

Arayüzden çağrılan sorgulardan en az biri “view” olarak tanımlanmış olmalıdır.

```
create or replace view sellers_loc as
select
    s.id as sellerid,
    s.seller_name as fname,
    s.seller_surname as lname,
    l.id as loc
from
    seller s,
    location l
where
    s.id = l.seller_id;
```

En az bir adet “sequence” oluşturmalı

```
create sequence if not exists productid_seq;
```

```
next_pid := nextval('productid_seq');
insert into product (id, product_name)
    values (next_pid, name);
productId := next_pid;
```

Sorgulardan en az birinde union veya intersect veya except.

```sql
select
    id
from
    product
except
select
    stock.product_id
from
    stock
where
    stock.seller_id = sellerId;
```

Sorguların en az biri aggregate fonksiyonlar içermeli, having ifadesi kullanılmalı.

```
select
    sale.product_id,
    sum(sale.quantity)::int
from
    sale
where
    sale.seller_id = sellerId
    and sale.sale_date >= starting
    and sale.sale_date <= ending
group by
    sale.product_id
having
    sum(sale.quantity) > 0;
```

Değerleri parametre olarak alıp ekrana sonuç döndüren 3 farklı SQL fonksiyon.
Bu fonksiyonlarda en az bir tane “record” ve “cursor” tanımı-kullanımı olmalıdır.

Record İçeren Fonksiyon

```
create type standrecord as (
        nmemb int,
        new_stand int
);
```

```
create or replace function add_user (username varchar(200), password varchar(200),
loc int, fname varchar(200), surname varchar(200))
    returns int
    as $$
declare
    stand standrecord;
    v_msg text;
begin
...
end;
$$
language 'plpgsql';
```

2. Fonksiyon

```
create or replace function list_products_not_in_stock (sellerId int)
    returns table ( product_id int, product_name varchar(200) )
    as $$
begin
    return query with not_taken as (
        select id
        from product
        except
        select stock.product_id
        from stock
        where stock.seller_id = sellerId )
    select p.id, p.product_name
    from product p, not_taken
    where not_taken.id = p.id;
end;
$$
language 'plpgsql';
```

Cursor İçeren Fonksiyon

```
create or replace function init_locations (n int)
    returns void
    as $$
declare
    v_msg text;
    declare loc_cur cursor for
        select
            *
        from
            generate_series(1, n);
begin
    for loc in loc_cur loop
        insert into location (id, seller_id)
            values (loc.generate_series, null);
    end loop;
end;

$$
language 'plpgsql';
```

2 adet trigger tanımlamalı ve arayüzden girilecek değerlerle tetiklemelisiniz.

```
create or replace function is_occupied_proc ()
    returns trigger
    as $$
begin
    if new.seller_id is null then
        return new;
    end if;
    if old.seller_id is not null then
        raise exception 'location is already occupied';
        return old;
    else
        return new;
    end if;
end;
$$
language 'plpgsql';
```

```
create or replace trigger is_occupied
before update on location
for each row execute procedure is_occupied_proc ();
```

2 adet trigger tanımlamalı ve arayüzden girilecek değerlerle tetiklemelisiniz.

```
create or replace function pos_quantity_proc ()
    returns trigger
    as $$
begin
    if old.quantity < 0 then
        raise exception 'quantity cannot be negative';
        return old;
    else
        return new;
    end if;
end;
$$
language 'plpgsql';
```

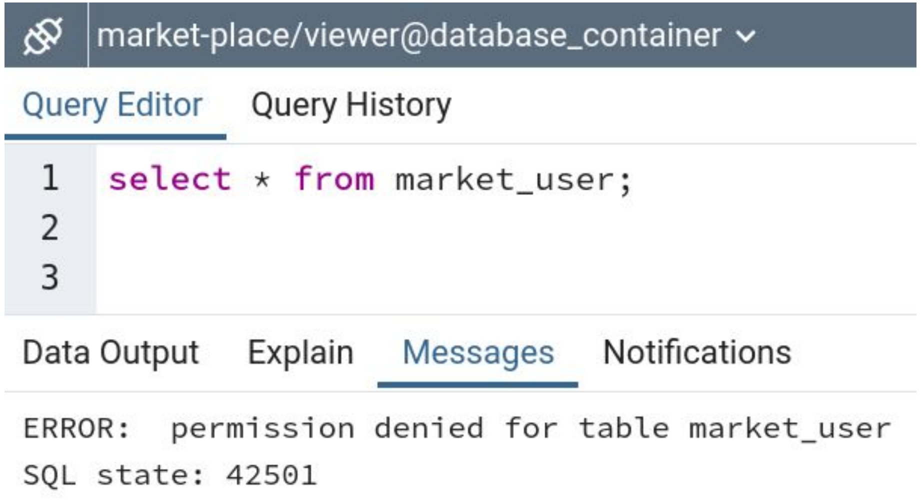
```
create or replace trigger pos_quantity
before update on stock
for each row execute procedure pos_quantity_proc ();
```

Rol ve Kullanıcılar

Rol Oluşturma Kodu

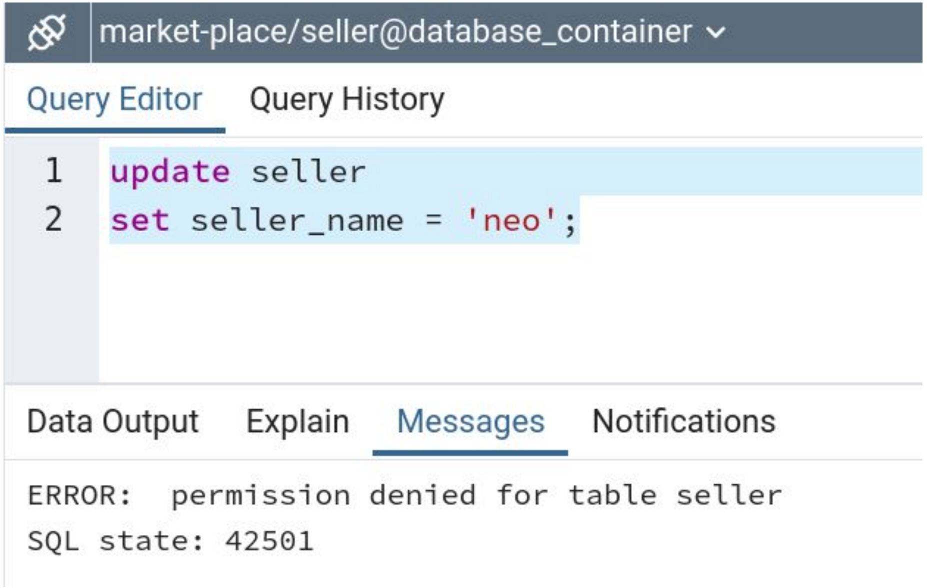
```
create role seller login password 'XtremelySaf3Pa5sworDs3LLer';
create role viewer login password 'typica1PassWorD';

/* SELLER PERMISSIONS */
grant usage on schema public to seller;
grant select on product, location, stock, sale to seller;
grant insert on sale, product, stock to seller;
grant update on stock to seller;
grant delete on stock to seller;
grant execute on function list_available_loc, list_products_not_in_stock, list_sales_usr, list_stock,
list_one_product, update_quantity, sell, add_product, add_stock, remove_stock to seller;

/* VIEWER PERMISSIONS */
grant usage on schema public to viewer;
grant select on stock, seller, location, product to viewer;
grant execute on function list_all_products, list_product_stocks to viewer;
```

Örnekler



```
select * from market_user;
```

```
ERROR:  permission denied for table market_user
SQL state: 42501
```



```
update seller
set seller_name = 'neo';
```

```
ERROR:  permission denied for table seller
SQL state: 42501
```