This task contains screenshots of solutions.

1. Connect to psql using superuser.

Then create user matt with password test valid end of this year.

```
psql -h localhost -U postgres -d mybusiness
```

“postgres” is the default superuser for a PostgreSQL database.

Notice the reticle being a hash, “#”, indicating a superuser, as opposed to

```
mybusiness=>
```

, which indicates that you are not logged in as a superuser.

```
mybusiness=# CREATE USER matt WITH PASSWORD 'test' VALID UNTIL '2022-12-31';
CREATE ROLE
mybusiness=# \du
                                   List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 dev       | Create DB                                                  | {}
 matt      | Password valid until 2022-12-31 00:00:00+02                | {}
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
```

2. Grant SELECT privilege for salesman table to matt

Grant CREATEUSER role to matt

Then quit the psql and then connect to your database as matt.

As matt find all salesman id and name

As matt try to find all customer names from customer table. What happens and why?

```
mybusiness=# GRANT SELECT ON salesman TO matt;
GRANT
```

```
mybusiness=# ALTER ROLE matt WITH CREATEROLE;
ALTER ROLE
```

Then logging in as matt:

```
psql -U matt -h localhost -d mybusiness
```

```
mybusiness=> SELECT salesman_id, name FROM salesman;
 salesman_id |    name
-------------+------------
        5001 | James Hoow
        5002 | Nail Knite
        5005 | Pit Alex
        5006 | Mc Lyon
        5003 | Lauson Hen
        5011 | Sam Lawson
        5009 | Paul White
(7 rows)
```

Then we try to select all the names from the customer table:

```
mybusiness=> SELECT cust_name FROM customer;
ERROR:  permission denied for table customer
```

This gives an error, since we have only given matt SELECT permission for the “salesman table”, but not the “customer” table.

3. Quit the psql and login as super user

Grant matt SELECT and DELETE privileges to customer and orders

Grant Matt UPDATE privilege to orders table

```
mybusiness=# GRANT SELECT, DELETE ON customer, orders TO matt;
GRANT
mybusiness=# GRANT UPDATE ON orders TO matt;
GRANT
```

4. As matt

Find all the customers (all columns) from customer table

Update order whose id is 70008 purch_amount to 5800

Create user lisa with password test

```
mybusiness=> SELECT * FROM customer;
 customer_id |   cust_name    |    city    | grade | salesman_id
-------------+----------------+------------+-------+-------------
        3002 | Nick Rimando   | New York   |   100 |        5001
        3007 | Brad Davis     | New York   |   200 |        5001
        3008 | Julian Green   | London     |   300 |        5002
        3004 | Fabian Johnson | Paris      |   300 |        5006
        3009 | Geoff Cameron  | Berlin     |   100 |        5003
        3001 | Brad Guzan     | London     |   300 |        5005
        3005 | Graham Zusi    | California |   300 |        5002
        3003 | Jozy Altidor   | Moscow     |   200 |        5009
(8 rows)
```

```
mybusiness=> UPDATE orders
mybusiness-> SET purch_amt = 5800
mybusiness-> WHERE ord_no = 70008;
UPDATE 1
```

```
mybusiness=> CREATE USER lisa WITH PASSWORD 'test';
CREATE ROLE
```

5. As matt

Create a group called staff

Add lisa and matt to staff

Why are you able to do all of this as matt?

As superuser Grant staff SELECT, UPDATE and INSERT privileges to all your database tables (salesman, customer, orders)

```
mybusiness=> CREATE GROUP staff;
CREATE ROLE
mybusiness=> ALTER GROUP staff ADD USER matt, lisa;
ALTER ROLE
```

Matt has permission to create roles, thus being able to create groups as well.

```
mybusiness=# GRANT INSERT ON customer, salesman, orders TO staff;
GRANT
mybusiness=# GRANT SELECT ON customer, salesman, orders TO staff;
GRANT
mybusiness=# GRANT UPDATE ON customer, salesman, orders TO staff;
GRANT
```

6. As lisa Write one SELECT, INSERT and UPDATE SQL command to one of the tables and check if that works. Try to delete the row you inserted before. What happens?

Logging in as lisa:

```
psql -U lisa -d mybusiness -h localhost
```

```
mybusiness=> SELECT * FROM customer;
 customer_id |   cust_name    |    city    | grade | salesman_id
-------------+----------------+------------+-------+-------------
        3002 | Nick Rimando   | New York   |   100 |        5001
        3007 | Brad Davis     | New York   |   200 |        5001
        3008 | Julian Green   | London     |   300 |        5002
        3004 | Fabian Johnson | Paris      |   300 |        5006
        3009 | Geoff Cameron  | Berlin     |   100 |        5003
        3001 | Brad Guzan     | London     |   300 |        5005
        3005 | Graham Zusi    | California |   300 |        5002
        3003 | Jozy Altidor   | Moscow     |   200 |        5009
(8 rows)
```

```
mybusiness=> INSERT INTO customer (customer_id, cust_name, city, grade, salesman_id)
mybusiness-> VALUES (3010, 'Kili Pukki', 'Nuorgam', 100, 5001);
INSERT 0 1
```

```
mybusiness=> UPDATE customer
mybusiness-> SET city = 'Hanko'
mybusiness-> WHERE customer_id = 3008;
UPDATE 1
```

```
mybusiness=> DELETE FROM customer
mybusiness-> WHERE cust_name = 'Kili Pukki';
ERROR:  permission denied for table customer
```

We didn't give DELETE permissions to the "staff" group, nor the "lisa" user.

7. Do the same as in the previous task but as matt. Is there any difference compared to what you did as lisa?

Again, switching back to matt:

```
psql -U matt -h localhost -d mybusiness
```

```
mybusiness=> SELECT * FROM customer;
 customer_id |   cust_name    |    city    | grade | salesman_id
-------------+----------------+------------+-------+-------------
        3002 | Nick Rimando   | New York   |   100 |        5001
        3007 | Brad Davis     | New York   |   200 |        5001
        3004 | Fabian Johnson | Paris      |   300 |        5006
        3009 | Geoff Cameron  | Berlin     |   100 |        5003
        3001 | Brad Guzan     | London     |   300 |        5005
        3005 | Graham Zusi    | California |   300 |        5002
        3003 | Jozy Altidor   | Moscow     |   200 |        5009
        3010 | Kili Pukki     | Nuorgam    |   100 |        5001
        3008 | Julian Green   | Hanko      |   300 |        5002
(9 rows)
```

```
mybusiness=> INSERT INTO customer (customer_id, cust_name, city, grade, salesman_id)
mybusiness-> VALUES(3011, 'Joulu Pukki', 'Vladivostok', 100, 5001);
INSERT 0 1
```

```
mybusiness=> UPDATE customer
mybusiness-> SET city = 'New York'
mybusiness-> WHERE customer_id = 3011;
UPDATE 1
```

```
mybusiness=> DELETE FROM customer
mybusiness-> WHERE cust_name = 'Joulu Pukki';
DELETE 1
```

Matt has DELETE permissions, given previously.

8. As superuser drop DELETE privilege from tables customer and orders from Matt. Try out that Matt cannot delete rows anymore. Drop user lisa.

Change of user, yet again:

```
psql -U postgres -d mybusiness -h localhost
```

```
mybusiness=# REVOKE DELETE
mybusiness-# ON customer, orders
mybusiness-# FROM matt;
REVOKE
```

Changing back to matt:

```
psql -U matt -h localhost -d mybusiness
```

```
mybusiness=> DELETE FROM customer
mybusiness-> WHERE cust_name = 'Kili Pukki';
ERROR:  permission denied for table customer
```

Then changing back to the superuser, we can drop the user “lisa”:

```
mybusiness=# DROP USER lisa;
DROP ROLE
```