ANKARA UNIVERSITY

COM258/2058

PROJECT_1

Due Date: 28.03.2022 11:59 p.m.

Lab Asistant: Res. Asst. Zeynep Yıldırım

Overview

Welcome to COM258/2058☺. Within the scope of this project, you are expected to get used to the use of PostgreSQL, open source object-relational database system. In addition, it is expected that you will be able to query all possible SQL commands and create your own queries according to the desired output.

In the project, Task A part is given for you to install and get used to the programs (You do not need to put a screenshot in the document.).The main part you need to do is Task B, Task C and Task D. For Task B, there should be full screenshots of each query in the document. Projects of those who write the query on paper or cut the image will not be evaluated. The query and the output of the query should be clearly visible inside the image. For Task C, the queries should be handwritten on a piece of paper and the photograph should be attached to the document. And lastly, for Task D you must upload the updated Project_1.py as StudentNumber_Project_1.py and add the output images to the document.

You must submit the project as a single file in the "StudentNumber.pdf" format and StudentNumber_Project_1.py file. The project is individual and each student is expected to do it himself. In projects that are taken from each other or that are understood to be duplicates, all projects sent by the student will not be evaluated and the student's overall lab grade will be considered "0".

Task A: Getting Set Up

Before starting, you need to download **PostgreSQL** program from the "https://www.postgresql.org/download/" and install it on your computer. For more detailed information, the documents in the "https://www.postgresql.org/docs/" file can be examined.

After installing, you can access the program from the command prompt as shown in Figure 1.

Important Note: If you have not added the path of the PostgreSQL program to the system paths, you must work in the file location of the "bin" folder the program when running it at the command prompt.

For connection, you need to type the following command.

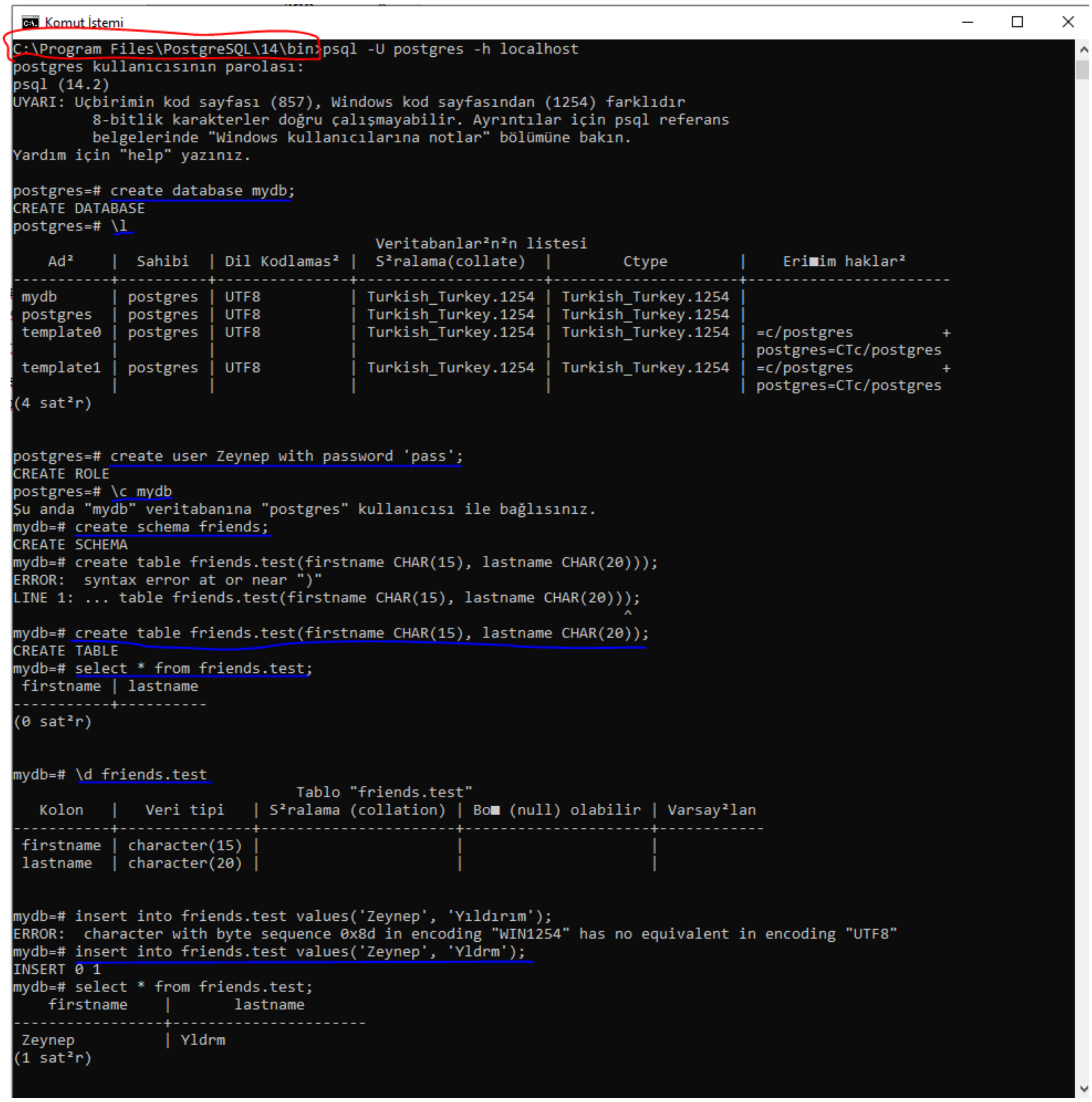
- psql –U postgres –h localhost

After running that command, the password you specified during installation will be asked. After this process, your database connection is made and you can see that you are currently running "postgres=#". Now it's time to create the database you will be working on.

- create database mydb;

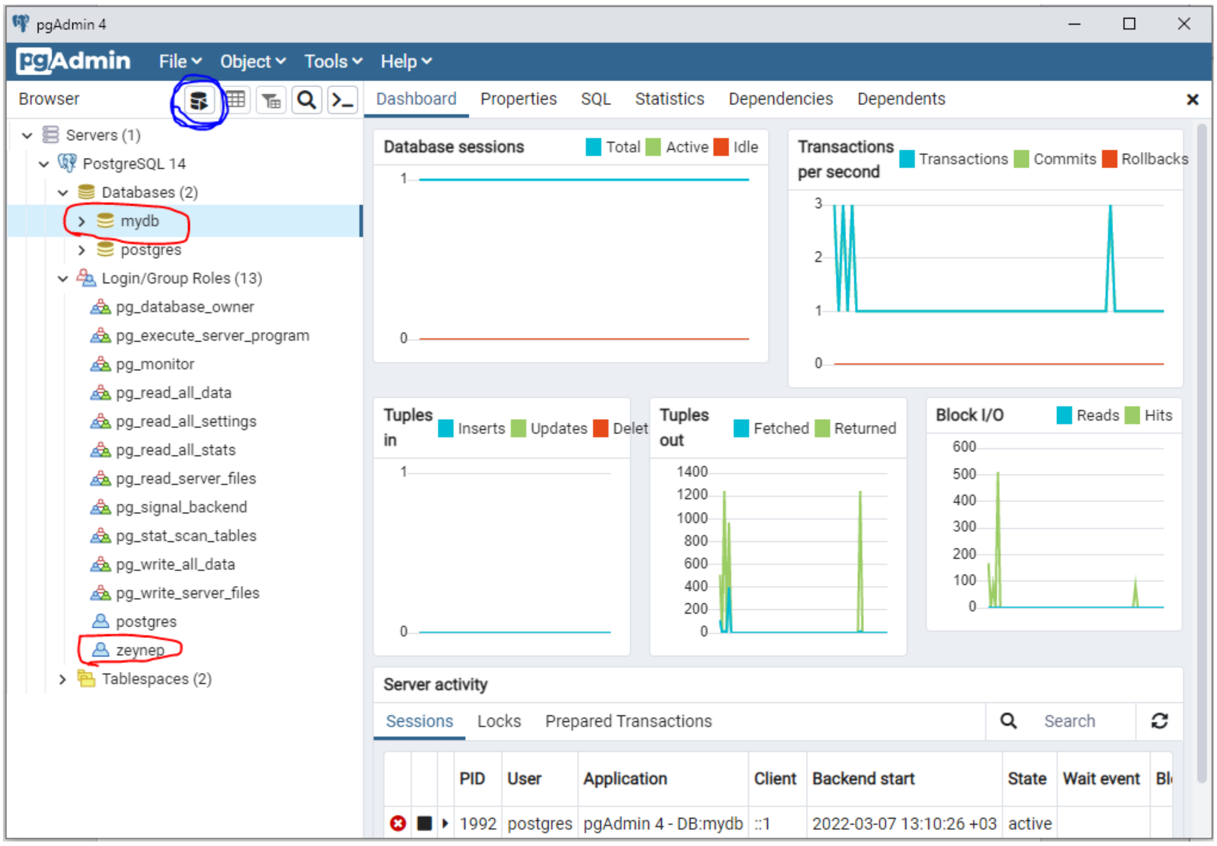
The "\l" command shows the existing databases, the "\c" command shows which user is connected to the database, the "\d" command shows the properties of the table. The "\q" command logs out of the system.

You can also use the **pgAdmin 4** program that comes by default in the installation instead of the command prompt. If it was not installed by default during installation, you can download it from the "https://www.pgadmin.org/download/" page. Of course, the pgAdmin program will also ask you for the password you set during installation for the connection. As you can see in Figure 2, all database operations on the command prompt screen are automatically detected by the pgadmin program. To run a query in the pgAdmin program, you must press the "Query Tool" button marked in blue in Figure 2.



```
C:\Program Files\PostgreSQL\14\bin>psql -U postgres -h localhost
postgres kullanıcısının parolası:
psql (14.2)
UYARI: Uçbirimin kod sayfası (857), Windows kod sayfasından (1254) farklıdır
         8-bitlik karakterler doğru çalışmayabilir. Ayrıntılar için psql referans
         belgelerinde "Windows kullanıcılarına notlar" bölümüne bakın.
Yardım için "help" yazınız.

postgres=# create database mydb;
CREATE DATABASE
postgres=# \l
                                         Veritabanlar²n²n listesi
    Ad²     |  Sahibi  | Dil Kodlamas² |  S²ralama(collate)  |        Ctype        |    Eri■im haklar²
-----------+----------+---------------+---------------------+---------------------+-----------------------
 mydb      | postgres | UTF8          | Turkish_Turkey.1254 | Turkish_Turkey.1254 |
 postgres  | postgres | UTF8          | Turkish_Turkey.1254 | Turkish_Turkey.1254 |
 template0 | postgres | UTF8          | Turkish_Turkey.1254 | Turkish_Turkey.1254 | =c/postgres          +
           |          |               |                     |                     | postgres=CTc/postgres
 template1 | postgres | UTF8          | Turkish_Turkey.1254 | Turkish_Turkey.1254 | =c/postgres          +
           |          |               |                     |                     | postgres=CTc/postgres
(4 sat²r)


postgres=# create user Zeynep with password 'pass';
CREATE ROLE
postgres=# \c mydb
Şu anda "mydb" veritabanına "postgres" kullanıcısı ile bağlısınız.
mydb=# create schema friends;
CREATE SCHEMA
mydb=# create table friends.test(firstname CHAR(15), lastname CHAR(20)));
ERROR:  syntax error at or near ")"
LINE 1: ... table friends.test(firstname CHAR(15), lastname CHAR(20)));
                                                                      ^
mydb=# create table friends.test(firstname CHAR(15), lastname CHAR(20));
CREATE TABLE
mydb=# select * from friends.test;
 firstname | lastname
-----------+----------
(0 sat²r)


mydb=# \d friends.test
                             Tablo "friends.test"
   Kolon   |   Veri tipi    | S²ralama (collation) | Bo■ (null) olabilir | Varsay²lan
-----------+---------------+----------------------+---------------------+------------
 firstname | character(15) |                      |                     |
 lastname  | character(20) |                      |                     |


mydb=# insert into friends.test values('Zeynep', 'Yıldırım');
ERROR:  character with byte sequence 0x8d in encoding "WIN1254" has no equivalent in encoding "UTF8"
mydb=# insert into friends.test values('Zeynep', 'Yldrm');
INSERT 0 1
mydb=# select * from friends.test;
    firstname    |       lastname
-----------------+----------------------
 Zeynep          | Yldrm
(1 sat²r)
```

Figure 1. PostgreSQL database connection example with command promt



- Servers (1)
  - PostgreSQL 14
    - Databases (2)
      - mydb
      - postgres
    - Login/Group Roles (13)
      - pg_database_owner
      - pg_execute_server_program
      - pg_monitor
      - pg_read_all_data
      - pg_read_all_settings
      - pg_read_all_stats
      - pg_read_server_files
      - pg_signal_backend
      - pg_stat_scan_tables
      - pg_write_all_data
      - pg_write_server_files
      - postgres
      - zeynep
    - Tablespaces (2)

PID	User	Application	Client	Backend start	State	Wait event	Bl
1992	postgres	pgAdmin 4 - DB:mydb	::1	2022-03-07 13:10:26 +03	active		

Figure 2. Example of PgAdmin 4

Task B: Querying - SQL!

Now, after getting a little familiar with the program, we come to the part where you have to do it!!! In this part, you need to import the attached **northwind.sql**. You can perform the queries on the command screen or in pgAdmin. This choice is up to you!

Attention!!! Each student should make the database name they create as their student number. Also, each student should create a schema named StudentNumber_Project1 and import the given .sql file in there.

You can see the database structure in Figure 3. You must examine the recorded data in each table before performing the queries. The given database is widely used. You can search the internet for your questions about the database.

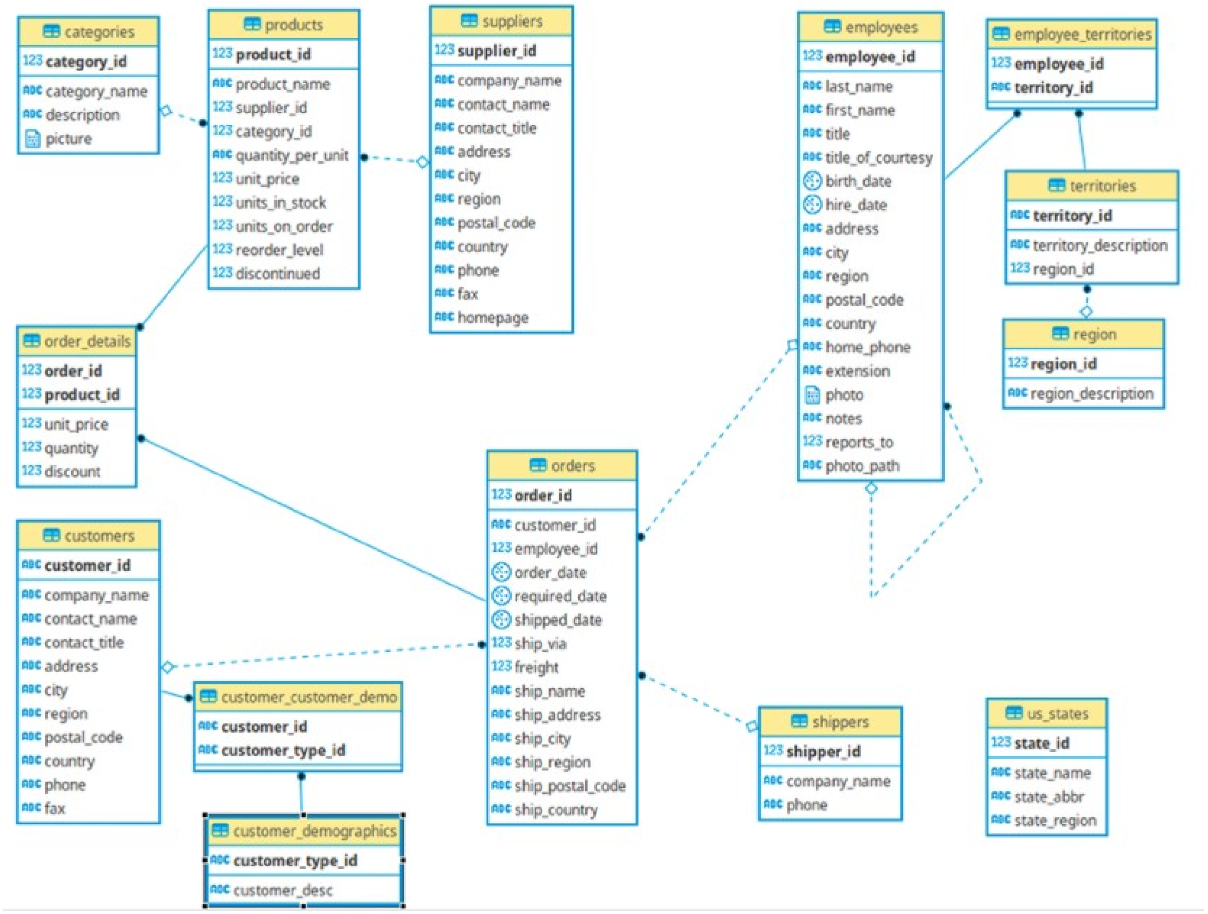
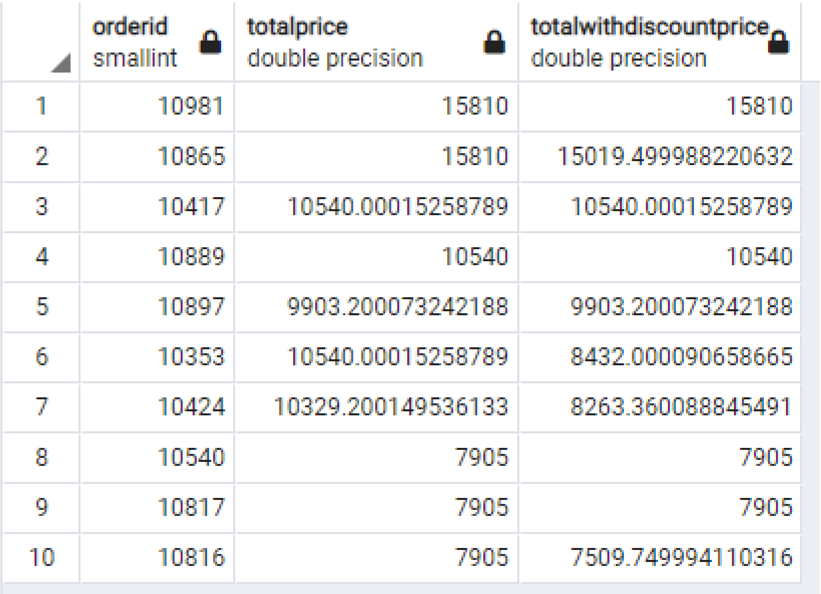


Figure 3. Nortwind Database

Write standard SQL queries to answer the following questions:

Attention!!! In order to get full marks, you need to obtain **the same query outputs** (pay attention to the column names and order). You are requested to specify the tables that you should use for queries. Therefore, it is expected that each student's query will be unique.

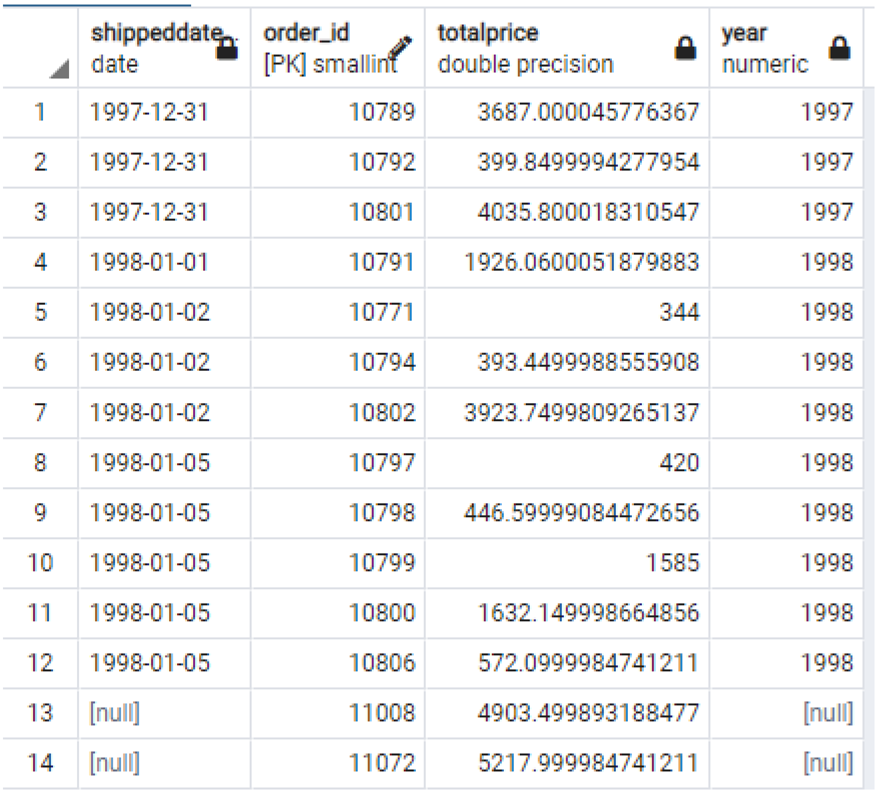
1. By finding the total price (unit_price*quantity) and discounted total prices (unit_price*quantity*(1-discount)) of the products, search for the top 10 when you sort them from high to low based on the discounted price calculation.



	orderid smallint	totalprice double precision	totalwithdiscountprice double precision
1	10981	15810	15810
2	10865	15810	15019.499988220632
3	10417	10540.00015258789	10540.00015258789
4	10889	10540	10540
5	10897	9903.200073242188	9903.200073242188
6	10353	10540.00015258789	8432.000090658665
7	10424	10329.200149536133	8263.360088845491
8	10540	7905	7905
9	10817	7905	7905
10	10816	7905	7509.749994110316

Figure 4. Question-1 Output

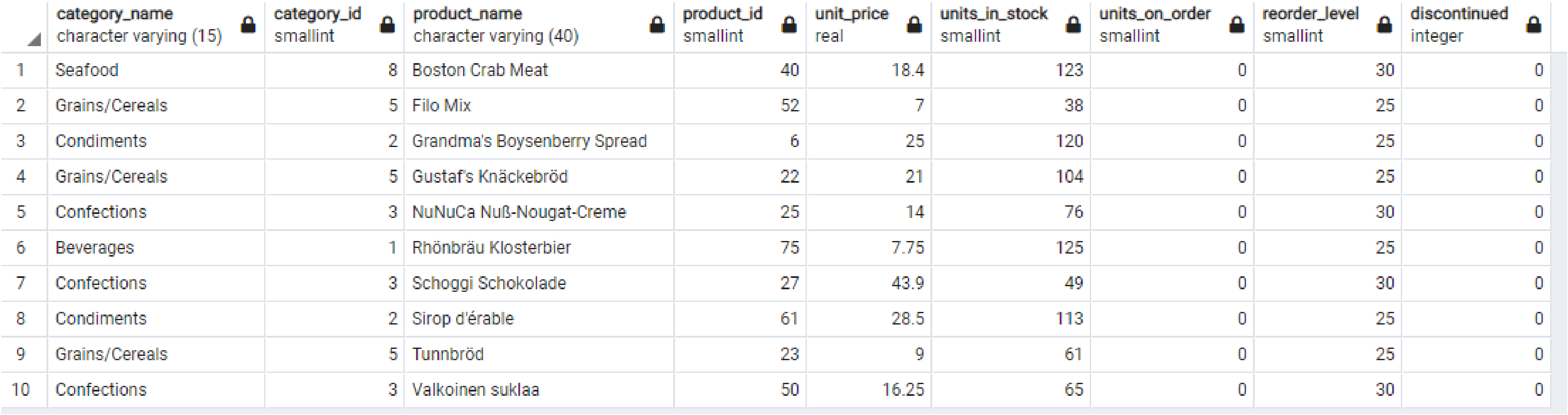
2. Query the total price of the products whose shipped_date value is between 1997-12-30 and 1998-1-5, and the data whose shipped_date value is null and the total price is greater than 4000.



	shippeddate date	order_id [PK] smallint	totalprice double precision	year numeric
1	1997-12-31	10789	3687.000045776367	1997
2	1997-12-31	10792	399.8499994277954	1997
3	1997-12-31	10801	4035.800018310547	1997
4	1998-01-01	10791	1926.0600051879883	1998
5	1998-01-02	10771	344	1998
6	1998-01-02	10794	393.4499988555908	1998
7	1998-01-02	10802	3923.7499809265137	1998
8	1998-01-05	10797	420	1998
9	1998-01-05	10798	446.59999084472656	1998
10	1998-01-05	10799	1585	1998
11	1998-01-05	10800	1632.149998664856	1998
12	1998-01-05	10806	572.0999984741211	1998
13	[null]	11008	4903.499893188477	[null]
14	[null]	11072	5217.999984741211	[null]

Figure 5. Question-2 Output

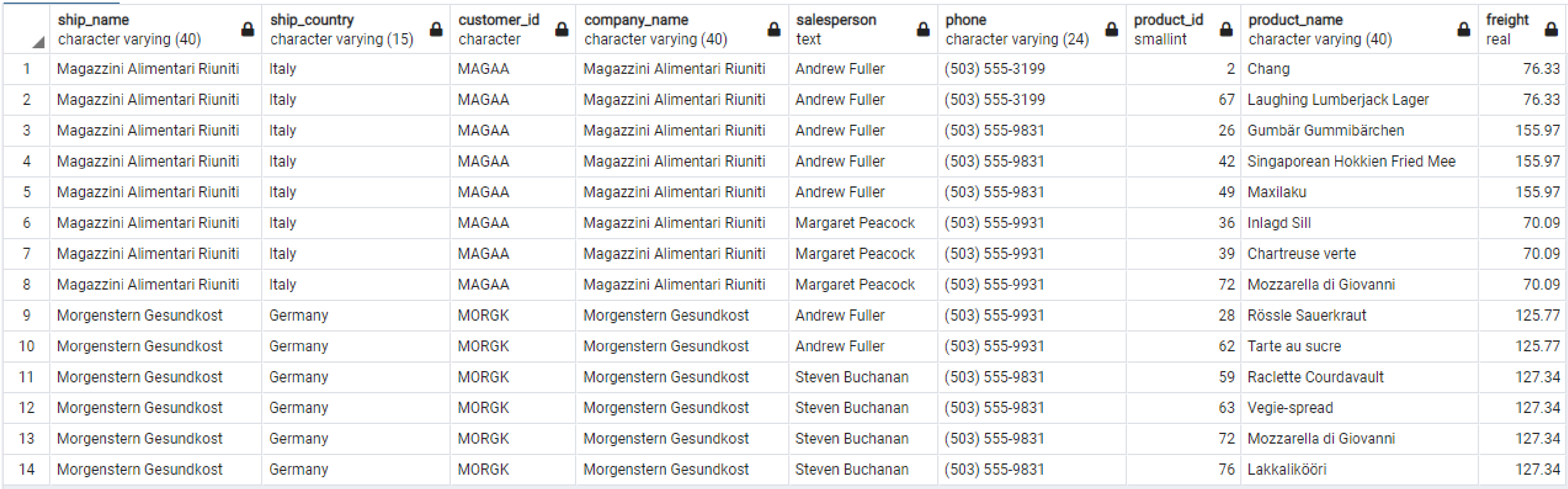
3. Query the data whose discontinued value is "0" and whose reorder_level value is higher than 20 and units_on_order value is 0.



	category_name character varying (15)	category_id smallint	product_name character varying (40)	product_id smallint	unit_price real	units_in_stock smallint	units_on_order smallint	reorder_level smallint	discontinued integer
1	Seafood	8	Boston Crab Meat	40	18.4	123	0	30	0
2	Grains/Cereals	5	Filo Mix	52	7	38	0	25	0
3	Condiments	2	Grandma's Boysenberry Spread	6	25	120	0	25	0
4	Grains/Cereals	5	Gustaf's Knäckebröd	22	21	104	0	25	0
5	Confections	3	NuNuCa Nuß-Nougat-Creme	25	14	76	0	30	0
6	Beverages	1	Rhönbräu Klosterbier	75	7.75	125	0	25	0
7	Confections	3	Schoggi Schokolade	27	43.9	49	0	30	0
8	Condiments	2	Sirop d'érable	61	28.5	113	0	25	0
9	Grains/Cereals	5	Tunnbröd	23	9	61	0	25	0
10	Confections	3	Valkoinen suklaa	50	16.25	65	0	30	0

Figure 6. Question-3 Output

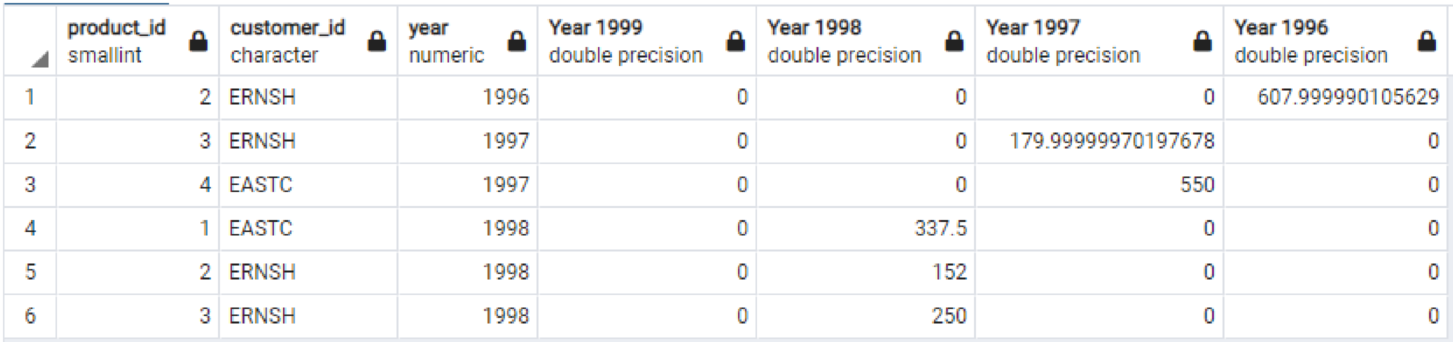
4. In the specified columns, query the data that ends with "y" for Ship_country, starts with "M" for customer_id, and whose freight value is greater than 70.



	ship_name character varying (40)	ship_country character varying (15)	customer_id character	company_name character varying (40)	salesperson text	phone character varying (24)	product_id smallint	product_name character varying (40)	freight real
1	Magazzini Alimentari Riuniti	Italy	MAGAA	Magazzini Alimentari Riuniti	Andrew Fuller	(503) 555-3199	2	Chang	76.33
2	Magazzini Alimentari Riuniti	Italy	MAGAA	Magazzini Alimentari Riuniti	Andrew Fuller	(503) 555-3199	67	Laughing Lumberjack Lager	76.33
3	Magazzini Alimentari Riuniti	Italy	MAGAA	Magazzini Alimentari Riuniti	Andrew Fuller	(503) 555-9831	26	Gumbär Gummibärchen	155.97
4	Magazzini Alimentari Riuniti	Italy	MAGAA	Magazzini Alimentari Riuniti	Andrew Fuller	(503) 555-9831	42	Singaporean Hokkien Fried Mee	155.97
5	Magazzini Alimentari Riuniti	Italy	MAGAA	Magazzini Alimentari Riuniti	Andrew Fuller	(503) 555-9831	49	Maxilaku	155.97
6	Magazzini Alimentari Riuniti	Italy	MAGAA	Magazzini Alimentari Riuniti	Margaret Peacock	(503) 555-9931	36	Inlagd Sill	70.09
7	Magazzini Alimentari Riuniti	Italy	MAGAA	Magazzini Alimentari Riuniti	Margaret Peacock	(503) 555-9931	39	Chartreuse verte	70.09
8	Magazzini Alimentari Riuniti	Italy	MAGAA	Magazzini Alimentari Riuniti	Margaret Peacock	(503) 555-9931	72	Mozzarella di Giovanni	70.09
9	Morgenstern Gesundkost	Germany	MORGK	Morgenstern Gesundkost	Andrew Fuller	(503) 555-9931	28	Rössle Sauerkraut	125.77
10	Morgenstern Gesundkost	Germany	MORGK	Morgenstern Gesundkost	Andrew Fuller	(503) 555-9931	62	Tarte au sucre	125.77
11	Morgenstern Gesundkost	Germany	MORGK	Morgenstern Gesundkost	Steven Buchanan	(503) 555-9831	59	Raclette Courdavault	127.34
12	Morgenstern Gesundkost	Germany	MORGK	Morgenstern Gesundkost	Steven Buchanan	(503) 555-9831	63	Vegie-spread	127.34
13	Morgenstern Gesundkost	Germany	MORGK	Morgenstern Gesundkost	Steven Buchanan	(503) 555-9831	72	Mozzarella di Giovanni	127.34
14	Morgenstern Gesundkost	Germany	MORGK	Morgenstern Gesundkost	Steven Buchanan	(503) 555-9831	76	Lakkalikööri	127.34

Figure 7. Question-4 Output

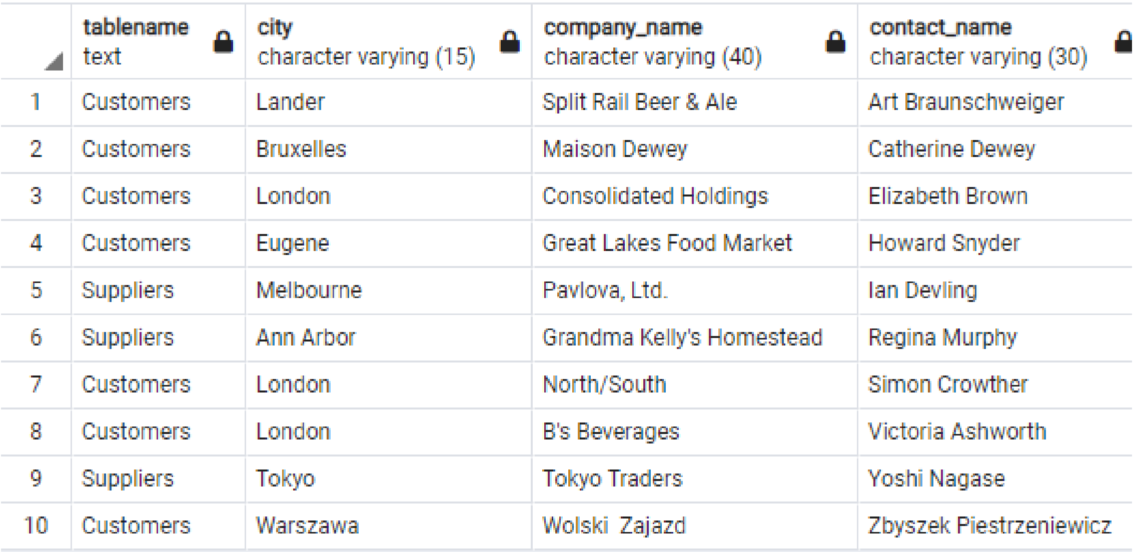
5. Write a query that calculates the total discounted price of all products by years (if there is no product, the value "0" will be written) and shows them by creating columns according to the years, with product_id less than 5 and customer_id starting with "E".



	product_id smallint	customer_id character	year numeric	Year 1999 double precision	Year 1998 double precision	Year 1997 double precision	Year 1996 double precision
1	2	ERNSH	1996	0	0	0	607.999990105629
2	3	ERNSH	1997	0	0	179.99999970197678	0
3	4	EASTC	1997	0	0	550	0
4	1	EASTC	1998	0	337.5	0	0
5	2	ERNSH	1998	0	152	0	0
6	3	ERNSH	1998	0	250	0	0

Figure 8. Question-5 Output

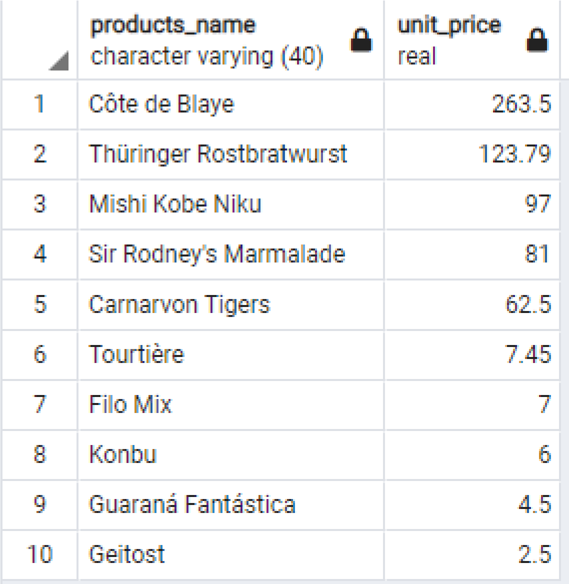
6. Combine the Customers and Suppliers tables and query the data containing the letter "w" in the contact_name in the costumer table and the letter "g" in the suppliers contact_name.



	tablename text	city character varying (15)	company_name character varying (40)	contact_name character varying (30)
1	Customers	Lander	Split Rail Beer & Ale	Art Braunschweiger
2	Customers	Bruxelles	Maison Dewey	Catherine Dewey
3	Customers	London	Consolidated Holdings	Elizabeth Brown
4	Customers	Eugene	Great Lakes Food Market	Howard Snyder
5	Suppliers	Melbourne	Pavlova, Ltd.	Ian Devling
6	Suppliers	Ann Arbor	Grandma Kelly's Homestead	Regina Murphy
7	Customers	London	North/South	Simon Crowther
8	Customers	London	B's Beverages	Victoria Ashworth
9	Suppliers	Tokyo	Tokyo Traders	Yoshi Nagase
10	Customers	Warszawa	Wolski Zajazd	Zbyszek Piestrzeniewicz

Figure 9. Question-6 Output

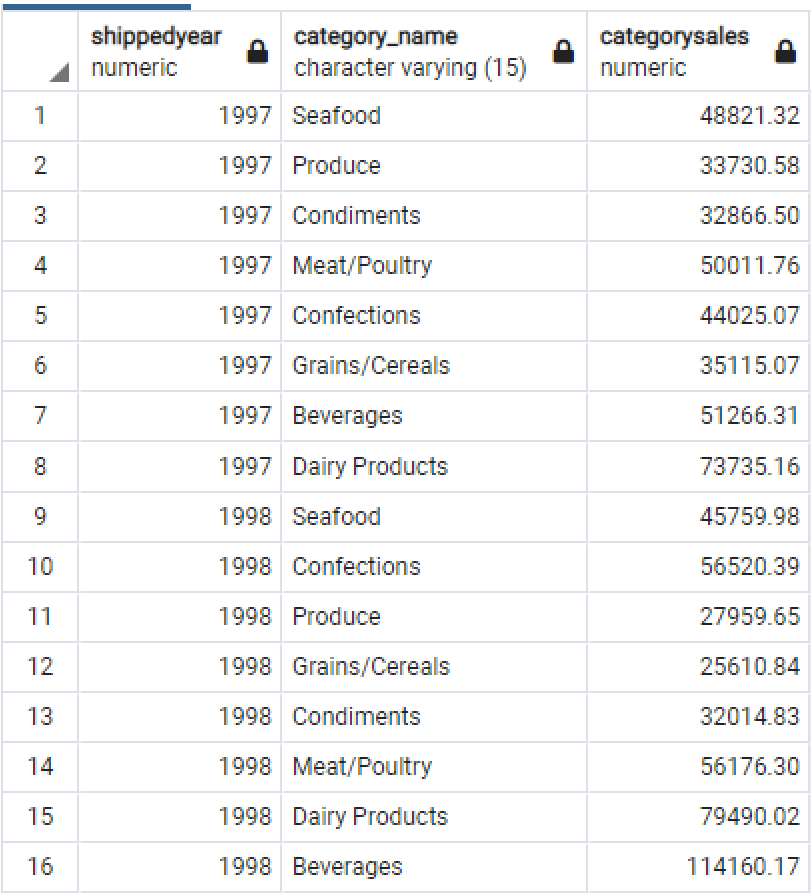
7. Show the first 5 products with the highest unit_price value and the first 5 products with the lowest, in order by unit_price.



	products_name character varying (40)	unit_price real
1	Côte de Blaye	263.5
2	Thüringer Rostbratwurst	123.79
3	Mishi Kobe Niku	97
4	Sir Rodney's Marmalade	81
5	Carnarvon Tigers	62.5
6	Tourtière	7.45
7	Filo Mix	7
8	Konbu	6
9	Guaraná Fantástica	4.5
10	Geitost	2.5

Figure 10. Question-7 Output

8. Query to sum of the sales made since June-1997 on a yearly basis according to the category_name. (Set to write two digits after the dot.)



	shippedyear numeric	category_name character varying (15)	categorysales numeric
1	1997	Seafood	48821.32
2	1997	Produce	33730.58
3	1997	Condiments	32866.50
4	1997	Meat/Poultry	50011.76
5	1997	Confections	44025.07
6	1997	Grains/Cereals	35115.07
7	1997	Beverages	51266.31
8	1997	Dairy Products	73735.16
9	1998	Seafood	45759.98
10	1998	Confections	56520.39
11	1998	Produce	27959.65
12	1998	Grains/Cereals	25610.84
13	1998	Condiments	32014.83
14	1998	Meat/Poultry	56176.30
15	1998	Dairy Products	79490.02
16	1998	Beverages	114160.17

Figure 11. Question-8 Output

Task C: Relational Algebra - SQL!

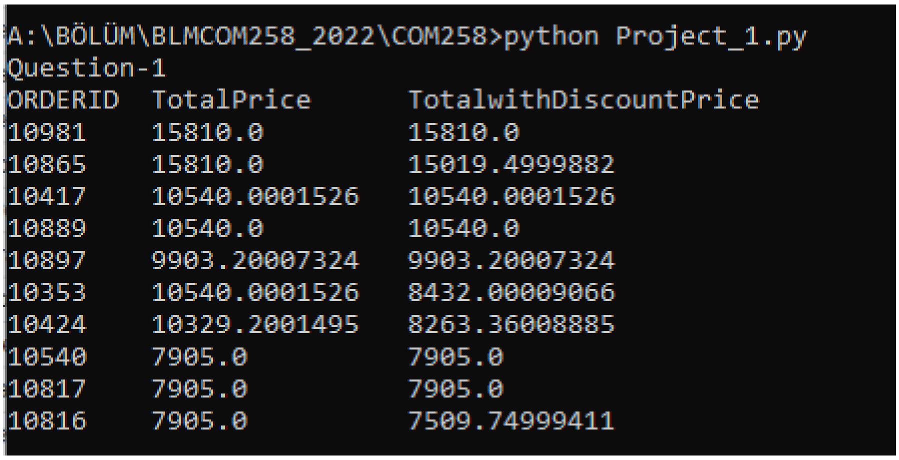
Now, you are expected to rewrite the SQL queries you wrote in Task B using relational algebra. As mentioned before, you are expected to write the solutions manually on a piece of paper and attach the photo to the document. Queries should be clearly visible.

Task D: Connect the Database in Python

In this part, you are expected to connect to the database you have created using python and execute the queries you have written. To connect to the database, you must first have python installed on your system. Then, you should then run the following command.

- python -m pip install psycopg2-binary

After the installation is complete, you need to run the Project_1.py file given to you by filling in the required places and add the screen outputs you received as in the image to your document (Figure 12). **You need to save the updated Project_1.py file as StudentNo_Project_1.py and upload it to the system.**



```
A:\BÖLÜM\BLMCOM258_2022\COM258>python Project_1.py
Question-1
ORDERID  TotalPrice       TotalwithDiscountPrice
10981    15810.0          15810.0
10865    15810.0          15019.4999882
10417    10540.0001526    10540.0001526
10889    10540.0          10540.0
10897    9903.20007324    9903.20007324
10353    10540.0001526    8432.00009066
10424    10329.2001495    8263.36008885
10540    7905.0           7905.0
10817    7905.0           7905.0
10816    7905.0           7509.74999411
```

Figure 12. Question-1 Output with Python