KONSTANTINOS PAGANOPOULOS 01769789

DIGITAL MARKETING ANALYTICS

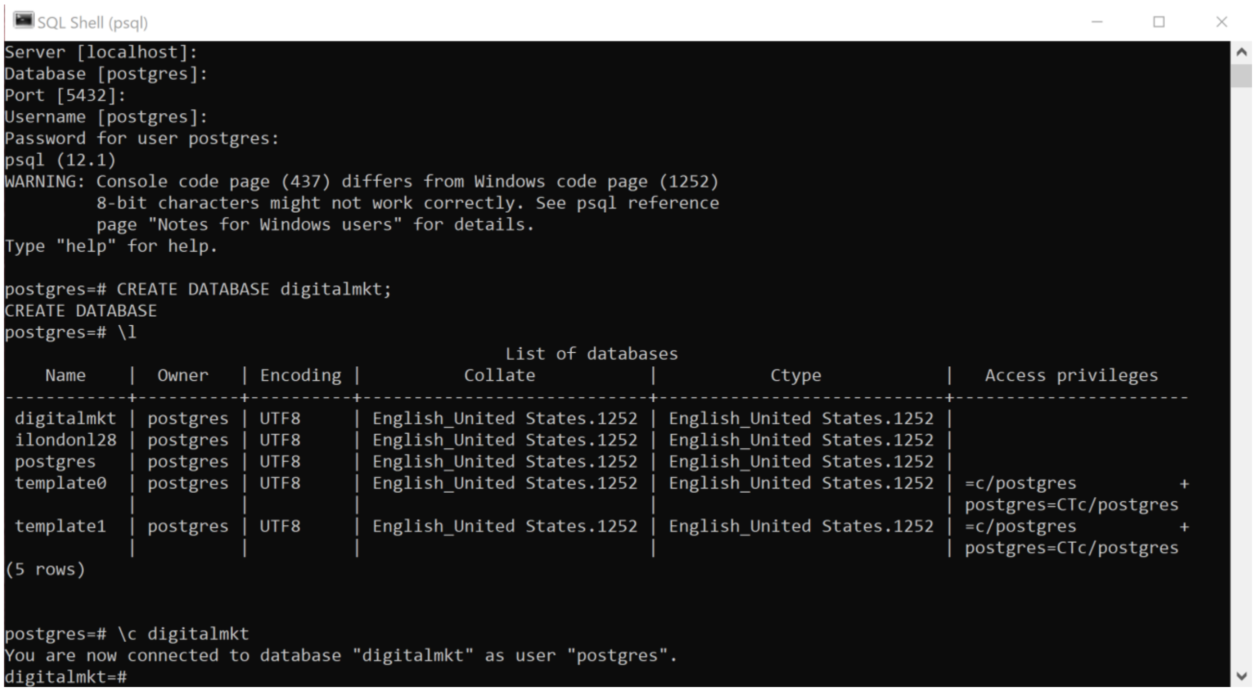
HW1 04/05/2020

Introduction

In the coursework assignment we will analyse a data set from a multichannel company with sales of several hundred million dollars per year, in order to produce some digital marketing insights. Various screenshots are taken through the whole procedure to help the reader follow easily (zoom in for more details).

Question 1

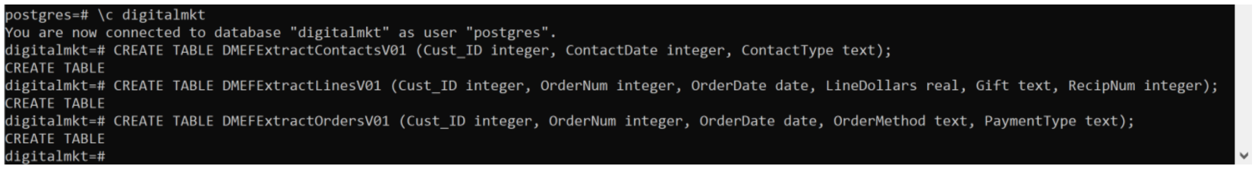
First of all we connect to our sercer and create a Postgresql database as follows:



```
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Password for user postgres:
psql (12.1)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# CREATE DATABASE digitalmkt;
CREATE DATABASE
postgres=# \l
                                                 List of databases
    Name    |  Owner   | Encoding |          Collate           |           Ctype            |   Access privileges
------------+----------+----------+----------------------------+----------------------------+-----------------------
 digitalmkt | postgres | UTF8     | English_United States.1252 | English_United States.1252 |
 ilondonl28 | postgres | UTF8     | English_United States.1252 | English_United States.1252 |
 postgres   | postgres | UTF8     | English_United States.1252 | English_United States.1252 |
 template0  | postgres | UTF8     | English_United States.1252 | English_United States.1252 | =c/postgres          +
            |          |          |                            |                            | postgres=CTc/postgres
 template1  | postgres | UTF8     | English_United States.1252 | English_United States.1252 | =c/postgres          +
            |          |          |                            |                            | postgres=CTc/postgres
(5 rows)


postgres=# \c digitalmkt
You are now connected to database "digitalmkt" as user "postgres".
digitalmkt=#
```

Screenshot 1: We create the "digitalmkt" database.

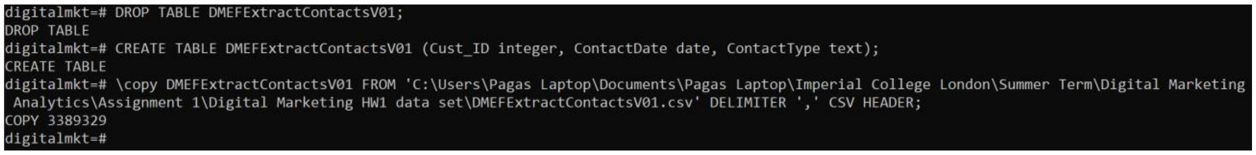
Then we create the 3 first tables that correspond to the following csv files: "DMEFExtractContactsV01", "DMEFExtractLinesV01" and "DMEFExtractOrdersV01". The remaining CSV has too many columns so we will load it with "SQLAlchemy" library of python.



```
postgres=# \c digitalmkt
You are now connected to database "digitalmkt" as user "postgres".
digitalmkt=# CREATE TABLE DMEFExtractContactsV01 (Cust_ID integer, ContactDate integer, ContactType text);
CREATE TABLE
digitalmkt=# CREATE TABLE DMEFExtractLinesV01 (Cust_ID integer, OrderNum integer, OrderDate date, LineDollars real, Gift text, RecipNum integer);
CREATE TABLE
digitalmkt=# CREATE TABLE DMEFExtractOrdersV01 (Cust_ID integer, OrderNum integer, OrderDate date, OrderMethod text, PaymentType text);
CREATE TABLE
digitalmkt=#
```

Screenshot 2: We create the "Contacts", "Lines" and "Orders" table.

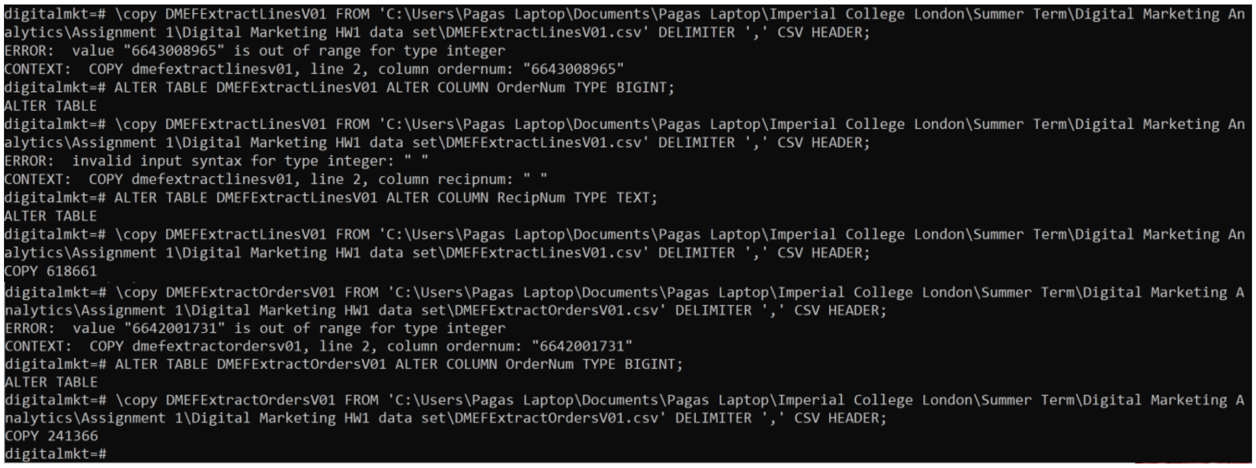
We then copy our csv files in the 3 tables. Note that we correct some mistakes that we have made. The mistakes are mainly about the type of the columns in the tables that we created.



```
digitalmkt=# DROP TABLE DMEFExtractContactsV01;
DROP TABLE
digitalmkt=# CREATE TABLE DMEFExtractContactsV01 (Cust_ID integer, ContactDate date, ContactType text);
CREATE TABLE
digitalmkt=# \copy DMEFExtractContactsV01 FROM 'C:\Users\Pagas Laptop\Documents\Pagas Laptop\Imperial College London\Summer Term\Digital Marketing
 Analytics\Assignment 1\Digital Marketing HW1 data set\DMEFExtractContactsV01.csv' DELIMITER ',' CSV HEADER;
COPY 3389329
digitalmkt=#
```

Screenshot 3: We recreate "Contacts" table because of the wrong type in ContactDate field and copy data from CSV.

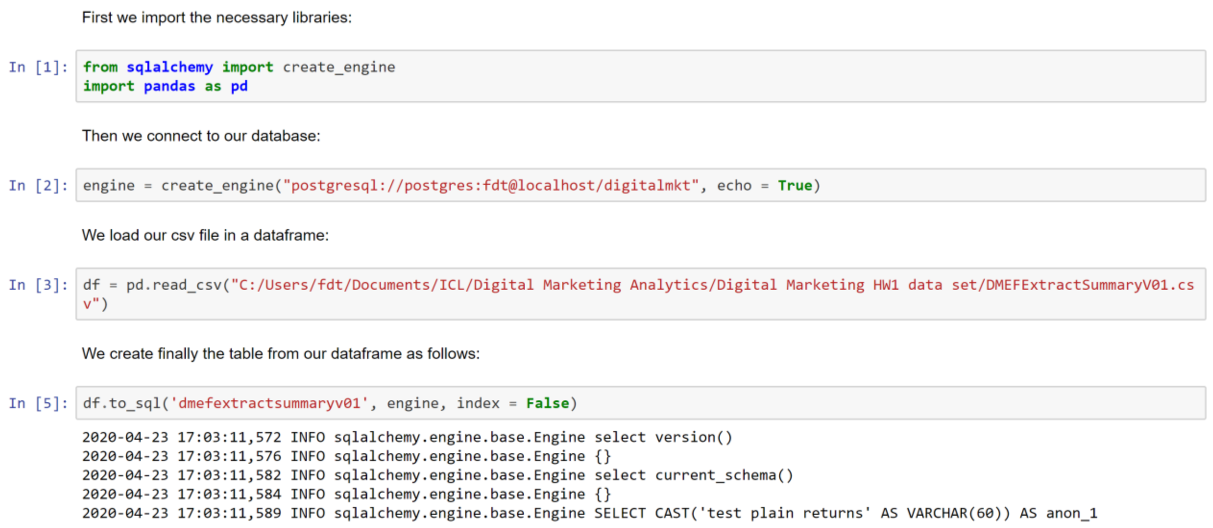
After that we copy the rest two files:



```
digitalmkt=# \copy DMEFExtractLinesV01 FROM 'C:\Users\Pagas Laptop\Documents\Pagas Laptop\Imperial College London\Summer Term\Digital Marketing An
alytics\Assignment 1\Digital Marketing HW1 data set\DMEFExtractLinesV01.csv' DELIMITER ',' CSV HEADER;
ERROR:  value "6643008965" is out of range for type integer
CONTEXT:  COPY dmefextractlinesv01, line 2, column ordernum: "6643008965"
digitalmkt=# ALTER TABLE DMEFExtractLinesV01 ALTER COLUMN OrderNum TYPE BIGINT;
ALTER TABLE
digitalmkt=# \copy DMEFExtractLinesV01 FROM 'C:\Users\Pagas Laptop\Documents\Pagas Laptop\Imperial College London\Summer Term\Digital Marketing An
alytics\Assignment 1\Digital Marketing HW1 data set\DMEFExtractLinesV01.csv' DELIMITER ',' CSV HEADER;
ERROR:  invalid input syntax for type integer: " "
CONTEXT:  COPY dmefextractlinesv01, line 2, column recipnum: " "
digitalmkt=# ALTER TABLE DMEFExtractLinesV01 ALTER COLUMN RecipNum TYPE TEXT;
ALTER TABLE
digitalmkt=# \copy DMEFExtractLinesV01 FROM 'C:\Users\Pagas Laptop\Documents\Pagas Laptop\Imperial College London\Summer Term\Digital Marketing An
alytics\Assignment 1\Digital Marketing HW1 data set\DMEFExtractLinesV01.csv' DELIMITER ',' CSV HEADER;
COPY 618661
digitalmkt=# \copy DMEFExtractOrdersV01 FROM 'C:\Users\Pagas Laptop\Documents\Pagas Laptop\Imperial College London\Summer Term\Digital Marketing A
nalytics\Assignment 1\Digital Marketing HW1 data set\DMEFExtractOrdersV01.csv' DELIMITER ',' CSV HEADER;
ERROR:  value "6642001731" is out of range for type integer
CONTEXT:  COPY dmefextractordersv01, line 2, column ordernum: "6642001731"
digitalmkt=# ALTER TABLE DMEFExtractOrdersV01 ALTER COLUMN OrderNum TYPE BIGINT;
ALTER TABLE
digitalmkt=# \copy DMEFExtractOrdersV01 FROM 'C:\Users\Pagas Laptop\Documents\Pagas Laptop\Imperial College London\Summer Term\Digital Marketing A
nalytics\Assignment 1\Digital Marketing HW1 data set\DMEFExtractOrdersV01.csv' DELIMITER ',' CSV HEADER;
COPY 241366
digitalmkt=#
```

Screenshot 4: We copy the data from the "Lines" and "Orders" CSVs.

Then we copy the data from the "Summary" CSV to our table through "SQLAlchemy". For more information please see the attached ".py" and ".html" files.



First we import the necessary libraries:

```
In [1]: from sqlalchemy import create_engine
        import pandas as pd
```

Then we connect to our database:

```
In [2]: engine = create_engine("postgresql://postgres:fdt@localhost/digitalmkt", echo = True)
```

We load our csv file in a dataframe:

```
In [3]: df = pd.read_csv("C:/Users/fdt/Documents/ICL/Digital Marketing Analytics/Digital Marketing HW1 data set/DMEFExtractSummaryV01.cs
        v")
```

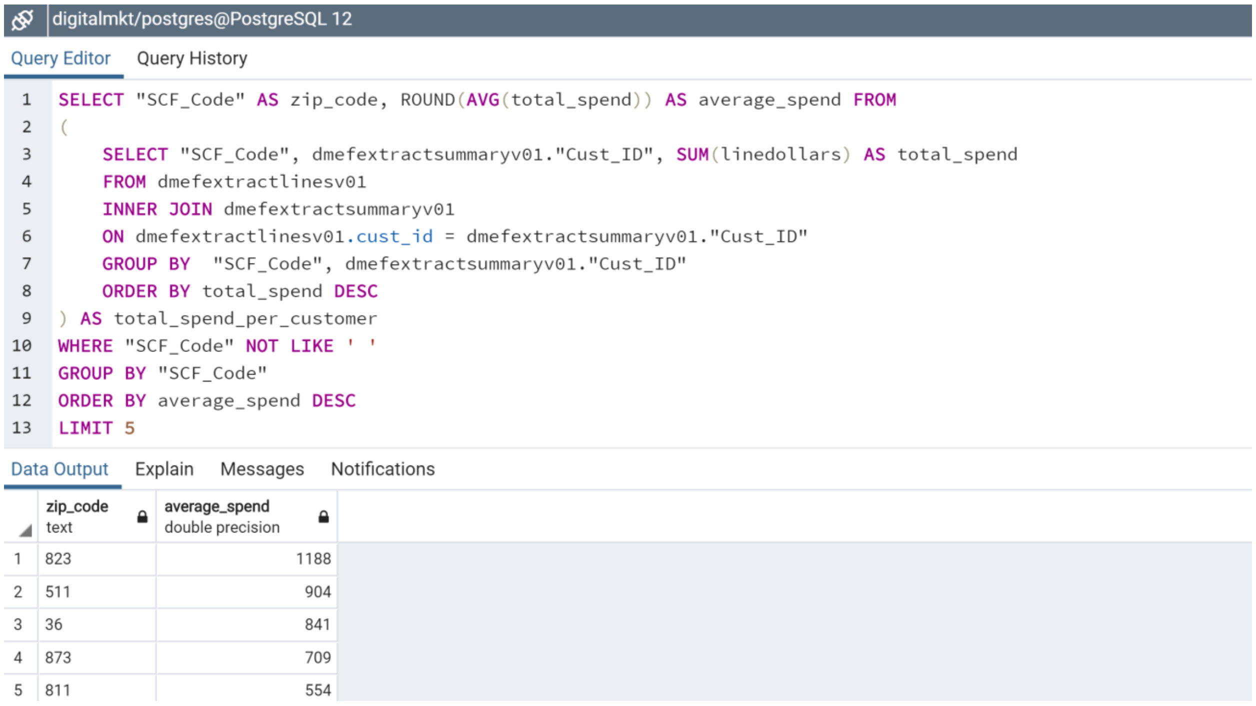
We create finally the table from our dataframe as follows:

```
In [5]: df.to_sql('dmefextractsummaryv01', engine, index = False)

        2020-04-23 17:03:11,572 INFO sqlalchemy.engine.base.Engine select version()
        2020-04-23 17:03:11,576 INFO sqlalchemy.engine.base.Engine {}
        2020-04-23 17:03:11,582 INFO sqlalchemy.engine.base.Engine select current_schema()
        2020-04-23 17:03:11,584 INFO sqlalchemy.engine.base.Engine {}
        2020-04-23 17:03:11,589 INFO sqlalchemy.engine.base.Engine SELECT CAST('test plain returns' AS VARCHAR(60)) AS anon_1
```

Screenshot 5: We copy the data from the "Summary" CSV.

We will now identify the top 5 customer locations by average spend. In order to do that we will first have to search for the terms location and spend. We can see that our most accurate location indicator is the field “SCF_Code”, which gives the first three digits of the customer’s ZIPCode and is in the “dmefextractsummaryv01” table. Note that “StoreDist” field is not an accurate indicator for location, as it gives the estimated distance from the customer’s address to the nearest company store in miles (e.g. equal distance if in north, south, east or west). We can also see that we have several fields for spend. We will now determine the average spend, by simply calculating the average of the “LineDollars” field of “dmefextractlinesv01” table. That column is the selling price of the line item in dollars, so by calculating the average of the ordered selling prices we calculate the average spend.

As a result, we will have to join table “dmefextractsummaryv01” with “dmefextractlinesv01” on their common column “cust_id”. Note that in our final query we “group by” location and “order by” average spend in descending order. We take our first 5 lines so us to meet the question’s needs. However, in order to calculate the average spend we will first have to sum the spend of each customer. We calculate that in a subquery. Our code and its output is the following:



```sql
SELECT "SCF_Code" AS zip_code, ROUND(AVG(total_spend)) AS average_spend FROM
(
    SELECT "SCF_Code", dmefextractsummaryv01."Cust_ID", SUM(linedollars) AS total_spend
    FROM dmefextractlinesv01
    INNER JOIN dmefextractsummaryv01
    ON dmefextractlinesv01.cust_id = dmefextractsummaryv01."Cust_ID"
    GROUP BY  "SCF_Code", dmefextractsummaryv01."Cust_ID"
    ORDER BY total_spend DESC
) AS total_spend_per_customer
WHERE "SCF_Code" NOT LIKE ' '
GROUP BY "SCF_Code"
ORDER BY average_spend DESC
LIMIT 5
```

	zip_code text	average_spend double precision
1	823	1188
2	511	904
3	36	841
4	873	709
5	811	554

Screenshot 6: SQL Query for question 1 and its output on pgAdmin.

Therefore, the locations with the 5 highest average spending in descending order are the ones with the following ZIP Codes: 823, 511, 36, 873, 811.

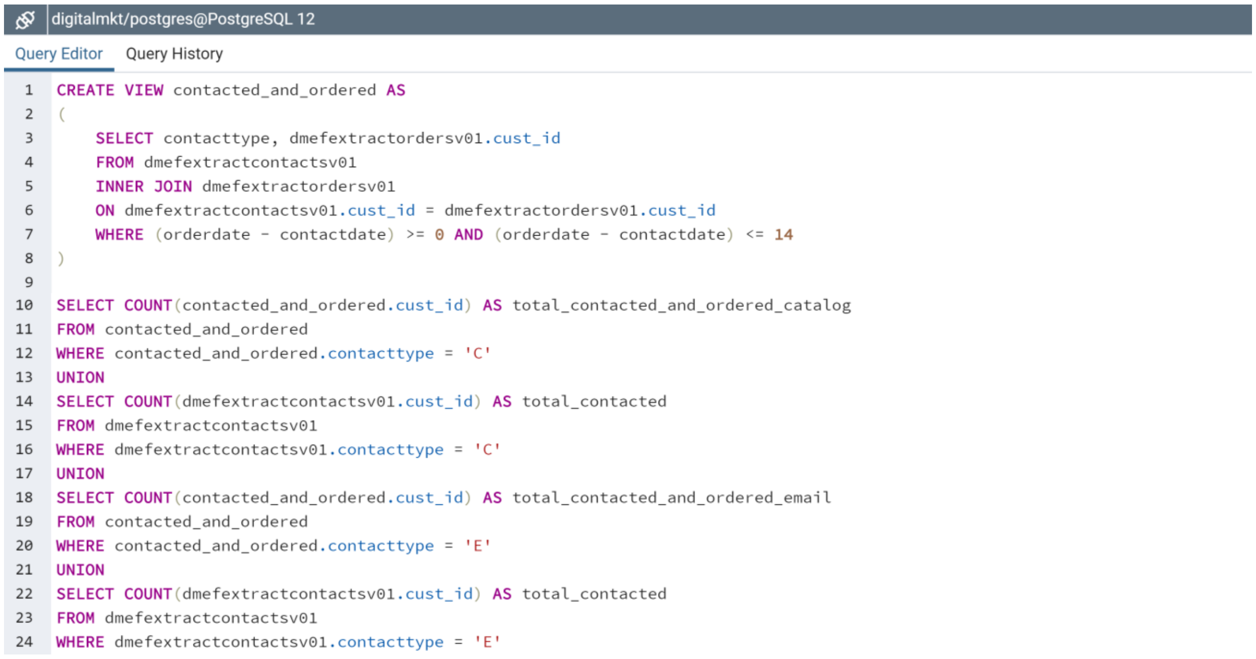
Also note that we rounded the spend result to no decimal digits and named the new column by “average_spend”. We also had to use some quotations for fields from “dmefextractsummaryv01” table. Maybe because we created that table with SQLAlchemy as previously shown and not with the “\copy” command. Note that we also selected only the ZIP Codes that are not missing (NOT LIKE ‘ ‘ command).

QUESTION 2

A.

We will now analyse the data set to gain "insights" into the effectiveness of the various direct-marketing channels, specifically catalog mailing vs. email. First, we will raise the question of which channel has better response rates, catalog mailing or email? In order to do that, we will have to join the "dmefextractcontactsv01" with the "dmefextractordersv01" table to see how many of the customers contacted have actually made an order. We will store that join in a "contacted_and_ordered" view. Note that we will not count the distinct customers, since one customer may have been contacted or ordered several times. Those culculations will be done for both channels ("C" for catalog mailing and "E" for email). Also note that we are interested in the orders of the last 2 weeks. In other words, we make the assumption that a contact is converted into an order only if the orderdate is equal to the contactdate or within the next 2 weeks of it.

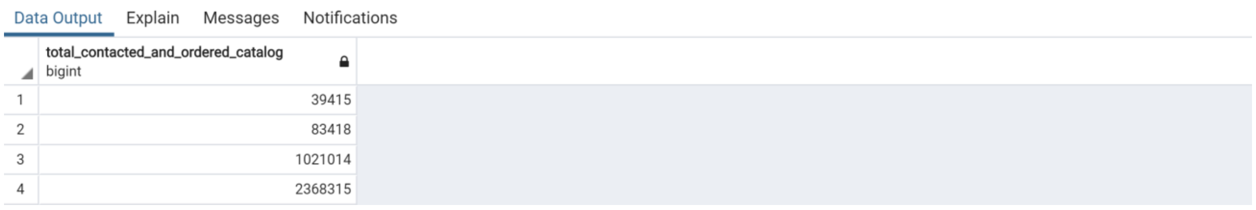
After that, we will divide those two number of total customers that have been contacted and made an order for each chanel by the total number of customers that have been contacted for each channel. Then we will have the two response rates for each of the two channels. Also note that we used the "UNION" command so as to combine the result sets of the 4 "SELECT" statements and remove duplicate rows between them. Our queries and outputs are the following:



```sql
CREATE VIEW contacted_and_ordered AS
(
    SELECT contacttype, dmefextractordersv01.cust_id
    FROM dmefextractcontactsv01
    INNER JOIN dmefextractordersv01
    ON dmefextractcontactsv01.cust_id = dmefextractordersv01.cust_id
    WHERE (orderdate - contactdate) >= 0 AND (orderdate - contactdate) <= 14
)

SELECT COUNT(contacted_and_ordered.cust_id) AS total_contacted_and_ordered_catalog
FROM contacted_and_ordered
WHERE contacted_and_ordered.contacttype = 'C'
UNION
SELECT COUNT(dmefextractcontactsv01.cust_id) AS total_contacted
FROM dmefextractcontactsv01
WHERE dmefextractcontactsv01.contacttype = 'C'
UNION
SELECT COUNT(contacted_and_ordered.cust_id) AS total_contacted_and_ordered_email
FROM contacted_and_ordered
WHERE contacted_and_ordered.contacttype = 'E'
UNION
SELECT COUNT(dmefextractcontactsv01.cust_id) AS total_contacted
FROM dmefextractcontactsv01
WHERE dmefextractcontactsv01.contacttype = 'E'
```

Screenshot 7: SQL Query for question 2A.

Therefore, based on the previous query's output we can now calculate the response rate for each channel as follows:


Data Output | Explain | Messages | Notifications

	total_contacted_and_ordered_catalog bigint
1	39415
2	83418
3	1021014
4	2368315

Screenshot 8: Output for first subquestion of question 2A.

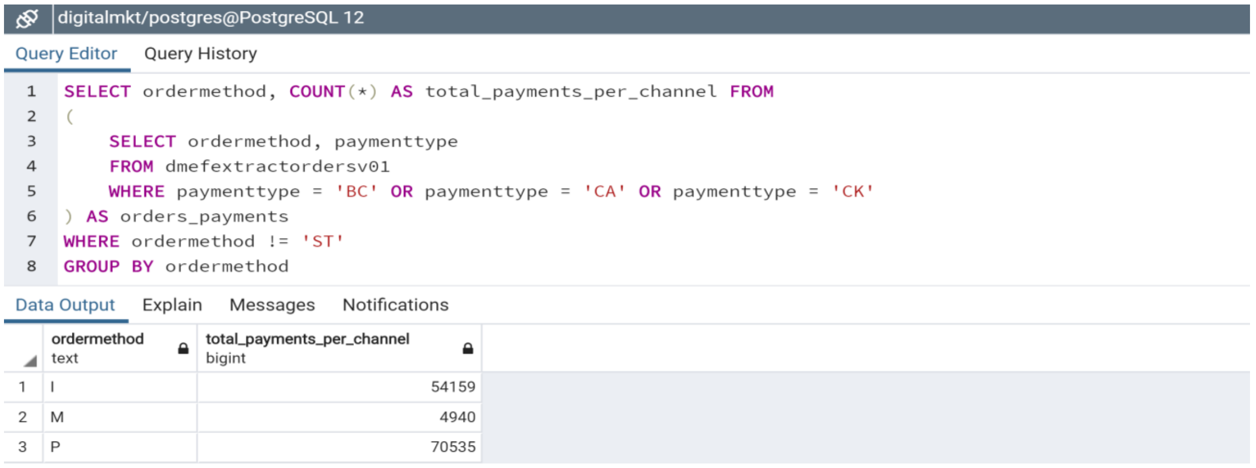
$$response\ rate_{catalog} = \frac{39,415}{1,021,014} \cong 3.8\%$$

$$response\ rate_{email} = \frac{83,418}{2,368,315} \cong 3.5\%$$

As a result, email has a better response rate.

Based on the previous logic, we can answer another, this time more simple, question regarding the effectiveness of the various direct-marketing channels. We can calculate how many payments occured in the catalog and email channel. We will only include the virtually most important payment types, since those are the ones who affect our sales the most. In other words, those of bankcard ('BC'), cash ('CA') or check ('CK'), since we want to compare how our two aforementioned marketing channels performed accross the most important payment types. That information about payments, in combination of the infromation about contacts from the previous question, will give us an indicator of how effective each of our direct-marketing channels was.

In order to calculate that, we will extract the information from the "dmefextractordersv01" table and count the total number of payments across our 3 most important payment types. We will present our results based on the group of channel they belong to. Note that we will not include the store channel ('ST'), since we focus on that of catalog mailing ("P", "M") and e-mail ("I").



```
SELECT ordermethod, COUNT(*) AS total_payments_per_channel FROM
(
    SELECT ordermethod, paymenttype
    FROM dmefextractordersv01
    WHERE paymenttype = 'BC' OR paymenttype = 'CA' OR paymenttype = 'CK'
) AS orders_payments
WHERE ordermethod != 'ST'
GROUP BY ordermethod
```

Data Output | Explain | Messages | Notifications

	ordermethod text	total_payments_per_channel bigint
1	I	54159
2	M	4940
3	P	70535

Screenshot 9: SQL Query and output for second subquestion of question 2A.

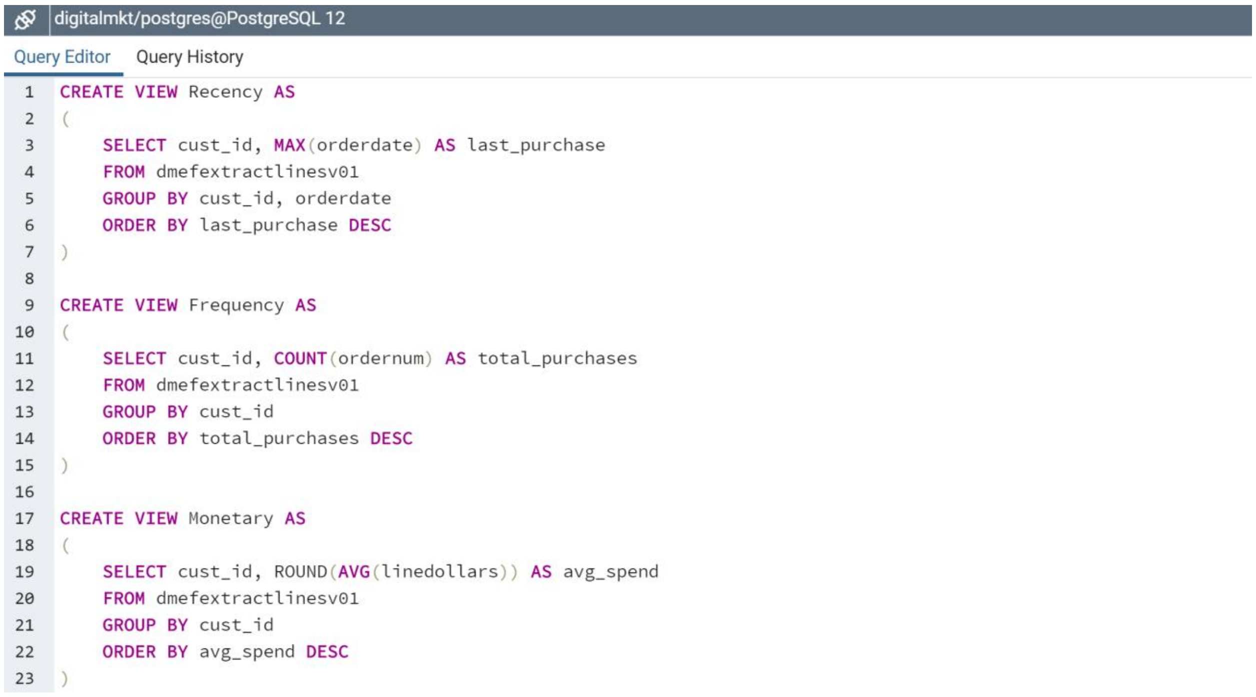
As a result, we can see that the total payments that occured via catalog-mailing (M+P) were 75,475 whereas those of e-mail (I) were 54,159. This result is expected as in the previous question we showed that the response rate of the catalog-mailing channels was greater than that of the e-mail, so we can conclude that the channel with the larger impact on the payments that were crucial in determining our sales was that of catalog-mailing and specificaly phone (70,535).

In conclusion, in both of the questions we saw that the company needs to improve its direct-marketing channels and especially that of e-mail, as it has a lower response rate, brings fewer payments to the company, and is the one in which the employees that are responsible for contact spend most time on (2,368,315).

B.

Now we will segment using the RFM dimensions (5 quantiles for each dimension). We will use the excel spreadsheet provided at the lectures.

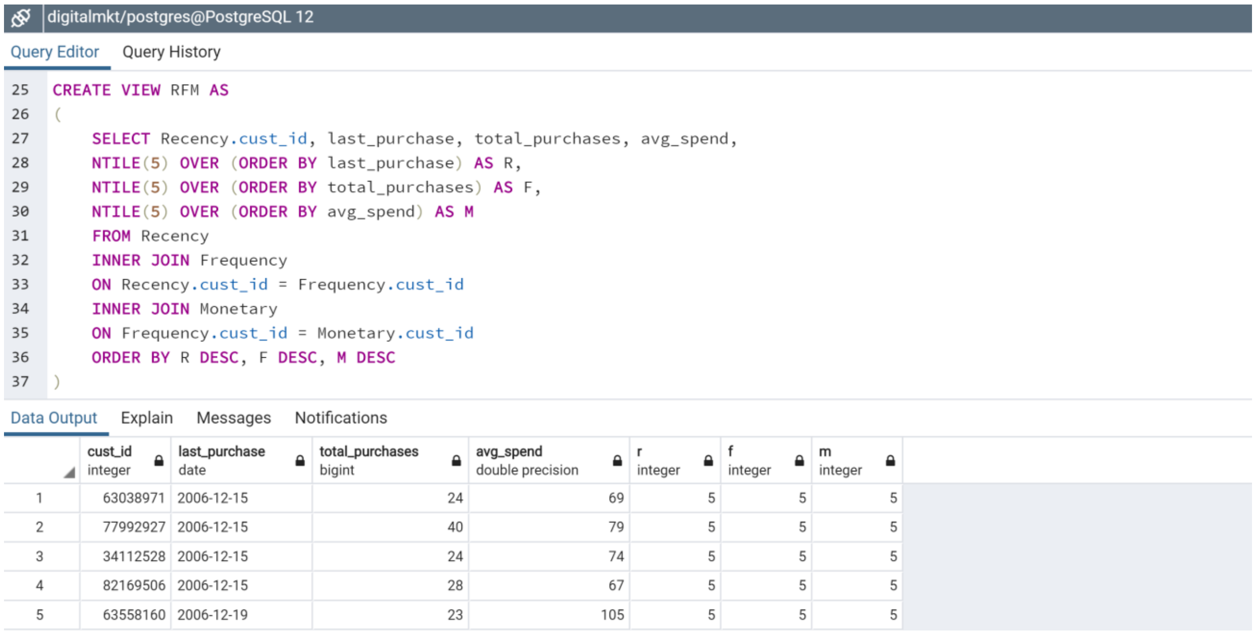
First, we create 3 views, in order to calculate for each customer the following fields: the day of its last purchase (recency), the total number of purchases (frequency) and the average spend (monetary). The SQL query is the following:



```
CREATE VIEW Recency AS
(
    SELECT cust_id, MAX(orderdate) AS last_purchase
    FROM dmefextractlinesv01
    GROUP BY cust_id, orderdate
    ORDER BY last_purchase DESC
)

CREATE VIEW Frequency AS
(
    SELECT cust_id, COUNT(ordernum) AS total_purchases
    FROM dmefextractlinesv01
    GROUP BY cust_id
    ORDER BY total_purchases DESC
)

CREATE VIEW Monetary AS
(
    SELECT cust_id, ROUND(AVG(linedollars)) AS avg_spend
    FROM dmefextractlinesv01
    GROUP BY cust_id
    ORDER BY avg_spend DESC
)
```

Screenshot 10: SQL Query for first subquestion of question 2B – RFM.

Now, we will create our final view as in the excel spreadsheet provided. Our final view will combine all previous 3 views, containing “cust_id”, “last_purchase”, “total_purchases”, “avg_spend”, as well as the distributed rows into a specified number of approximately equal groups, or buckets. This will be implemented through “NTILE” function as follows:


digitalmkt/postgres@PostgreSQL 12

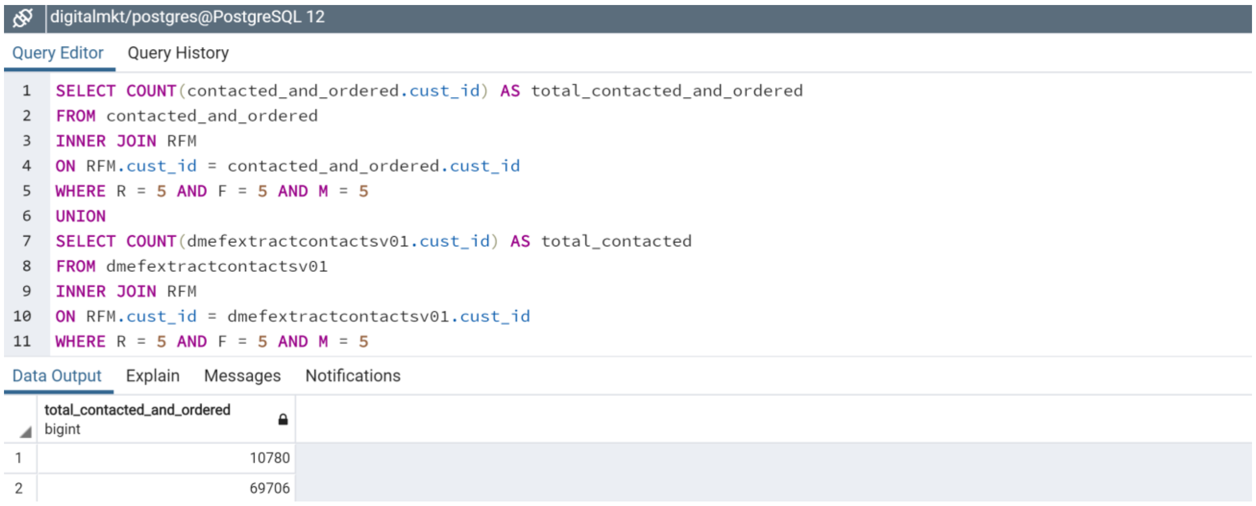
```sql
CREATE VIEW RFM AS
(
    SELECT Recency.cust_id, last_purchase, total_purchases, avg_spend,
    NTILE(5) OVER (ORDER BY last_purchase) AS R,
    NTILE(5) OVER (ORDER BY total_purchases) AS F,
    NTILE(5) OVER (ORDER BY avg_spend) AS M
    FROM Recency
    INNER JOIN Frequency
    ON Recency.cust_id = Frequency.cust_id
    INNER JOIN Monetary
    ON Frequency.cust_id = Monetary.cust_id
    ORDER BY R DESC, F DESC, M DESC
)
```

	cust_id integer	last_purchase date	total_purchases bigint	avg_spend double precision	r integer	f integer	m integer
1	63038971	2006-12-15	24	69	5	5	5
2	77992927	2006-12-15	40	79	5	5	5
3	34112528	2006-12-15	24	74	5	5	5
4	82169506	2006-12-15	28	67	5	5	5
5	63558160	2006-12-19	23	105	5	5	5

Screenshot 11: SQL Query and output for first subquestion of question 2B – RFM.

Note that we have segmented into 5 quantiles for each dimensions (5, 4, 3, 2, 1). As a result, we have in total 5 x 5 x 5 = 125 segments.

Now, for each one of those 125 segments we will calculate the response rate as we did in question 2A. Note that we make again use of the "contacted_and_ordered" view of question 2A. For simplicity reasons we do not provide again the "CREATE VIEW" statement. The extra query that we have to write is the following:


digitalmkt/postgres@PostgreSQL 12

```sql
SELECT COUNT(contacted_and_ordered.cust_id) AS total_contacted_and_ordered
FROM contacted_and_ordered
INNER JOIN RFM
ON RFM.cust_id = contacted_and_ordered.cust_id
WHERE R = 5 AND F = 5 AND M = 5
UNION
SELECT COUNT(dmefextractcontactsv01.cust_id) AS total_contacted
FROM dmefextractcontactsv01
INNER JOIN RFM
ON RFM.cust_id = dmefextractcontactsv01.cust_id
WHERE R = 5 AND F = 5 AND M = 5
```

	total_contacted_and_ordered bigint
1	10780
2	69706

Screenshot 12: SQL Query and output for first subquestion of question 2B – RFM and respond rates.

Note that in order to calculate the response rates of each of the 125 segments, we ran the above query changing the values "R", "F", and "M" other 124 times. For simplicity reasons, we provided only the query of the segment "555". Note that we could have avoided all those calculations by grouping by all segments.

For simplicity reasons we did not change the query again. From the above query we can calculate the response rate of that segment as follows:

Response rate (555) = 10,780/69,706 = 15.4%

We now calculate the rest of the response rates:

Response rate (554) = 11,857 / 69,730 = 17%, Response rate (553) = 13,826 / 82,241 = 16.8%,
Response rate (552) = 37,979 / 128,221 = 29.6%, Response rate (551) = 18,720 / 74,725 = 25%,
Response rate (545) = 12,241 / 105,579 = 11.5%, Response rate (544) = 16,304 / 129,935 = 12.5%,
Response rate (543) = 12,859 / 110,870 = 11.5%, Response rate (542) = 4,810 / 46,707 = 10.2%,
Response rate (541) = 2,045 / 24,609 = 8.3%, Response rate (535) = 12,956 / 131,534 = 9.8%,
Response rate (533) = 6,734 / 84,162 = 8%, Response rate (532) = 1,852 / 28,039 = 6.6%,
Response rate (531) = 580 / 14,608 = 3.9%, Response rate (525) = 8,408 / 107,155 = 7.8%,
Response rate (524) = 11,586 / 153,784 = 7.5%, Response rate (523) = 4428 / 69747 = 6%,
Response rate (522) = 348 / 8,074 = 4.3%, Response rate (521) = 141 / 5,916 = 2.3%,
Response rate (515) = 3,173 / 82,937 = 3.8%, Response rate (514) = 1,351 / 49,969 = 2.7%,
Response rate (513) = 1,510 / 44,130 = 3.4%, Response rate (512) = 64 / 3,939 = 1.6%,
Response rate (511) = 44 / 3,233 = 1.3%, Response rate (455) = 11,450 / 80,561 = 14.2%,
Response rate (454) = 12,812 / 88,927 = 14.4%, Response rate (453) = 18,083 / 11,2991 = 1.6%,
Response rate (452) = 38,038 / 141,672 = 26.8%, Response rate (451) = 18,920 / 83,762 = 22.5%,
Response rate (445) = 14,049 / 137,815 = 10.1%, Response rate (444) = 19,404 / 167,415 = 11.5%,
Response rate (443) = 15,386 / 149,417 = 10.2%, Response rate (442) = 4,013 / 60,150 = 6.6%,
Response rate (441) = 1,509 / 27,972 = 5.3%, Response rate (435) = 13,461 / 165,478 = 8.1%,
Response rate (434) = 14,020 / 173,072 = 8.1%, Response rate (433) = 8,894 / 125,807 = 7%,
Response rate (432) = 1,445 / 36,511 = 3.9%, Response rate (431) = 338 / 14,661 = 2.3%,
Response rate (425) = 7,632 / 141,919 = 5.3%, Response rate (424) = 10,198 / 205,322 = 4.9%,
Response rate (423) = 5,403 / 116,549 = 4.6%, Response rate (422) = 442 / 16,165 = 2.7%,
Response rate (421) = 56 / 5,557 = 1%, Response rate (415) = 2,023 / 151,254 = 1.3%,
Response rate (414) = 807 / 86,044 = 0.9%, Response rate (413) = 1,335 / 91,577 = 1.4%,
Response rate (412) = 117 / 13,888 = 0.8%, Response rate (411) = 35 / 4,091 = 0.8%,
Response rate (355) = 11,280 / 90,593 = 12.4%, Response rate (354) = 12,187 / 86,139 = 14.1%,
Response rate (353) = 17,735 / 119,354 = 14.8%, Response rate (352) = 33,270 / 156,964 = 21.1%,
Response rate (351) = 18,444 / 82,644 = 22.3%, Response rate (345) = 11,365 / 138,252 = 8.2%,
Response rate (344) = 15,169 / 162,146 = 9.3%, Response rate (343) = 14,356 / 164,843 = 8.7%,

Response rate (342) = 5,551 / 86,211 = 6.4%, Response rate (341) = 1,271 / 28,032 = 5.8%,
Response rate (335) = 9,390 / 160,109 = 5.8%, Response rate (334) = 10,541 / 168,426 = 6.2%,
Response rate (333) = 7,614 / 140,995 = 5.4%, Response rate (332) = 1,402 / 42,884 = 3.2%,
Response rate (331) = 436 / 15,147 = 2.8%, Response rate (325) = 3,997 / 123,284 = 3.2%,
Response rate (324) = 5,974 / 184,358 = 3.2%, Response rate (323) = 3,732 / 115,824 = 3.2%,
Response rate (322) = 263 / 18,289 = 1.4%, Response rate (321) = 73 / 5,511 = 1.3%,
Response rate (315) = 803 / 136,520 = 0.5%, Response rate (314) = 501 / 85,177 = 0.5%,
Response rate (313) = 877 / 100,995 = 0.8%, Response rate (312) = 71 / 19,788 = 0.3%,
Response rate (311) = 24 / 3,565 = 0.6%, Response rate (255) = 9,511 / 89,060 = 10.6%,
Response rate (254) = 11,806 / 98,051 = 12%, Response rate (253) = 15,215 / 118,950 = 12.7%,
Response rate (252) = 32,006 / 170,880 = 18.7%, Response rate (251) = 19,351 / 100,931 = 19.1%,
Response rate (245) = 7,999 / 130,433 = 6.1%, Response rate (244) = 10,582 / 142,786 = 7.4%,
Response rate (243) = 8,860 / 141,906 = 6.2%, Response rate (242) = 2,400 / 66,574 = 3.6%,
Response rate (241) = 954 / 33,998 = 2.8%, Response rate (235) = 4,688 / 117,691 = 3.9%,
Response rate (234) = 5,410 / 137,993 = 3.9%, Response rate (233) = 3,647 / 104,918 = 3.4%,
Response rate (232) = 641 / 39,295 = 1.6%, Response rate (231) = 151 / 16,610 = 0.9%,
Response rate (225) = 1,393 / 90,754 = 1.5%, Response rate (224) = 2,400 / 135,698 = 1.7%,
Response rate (223) = 1,365 / 91,963 = 1.4%, Response rate (222) = 71 / 17,988 = 0.3%,
Response rate (221) = 37 / 8,984 = 0.4%, Response rate (215) = 267 / 94,188 = 0.2%,
Response rate (214) = 85 / 55,092 = 0.1%, Response rate (213) = 199 / 75,480 = 0.2%,
Response rate (212) = 4 / 13,050 = 0.03%, Response rate (211) = 2 / 7,743 = 0.02%,
Response rate (155) = 11,071 / 124,240 = 8.9%, Response rate (154) = 14,480 / 127,794 = 11.3%,
Response rate (153) = 12,192 / 112,831 = 10.8%, Response rate (152) = 11,593 / 69,610 = 16.6%,
Response rate (151) = 4,267 / 22,198 = 19.2%, Response rate (145) = 10,858 / 191,485 = 5.6%,
Response rate (144) = 11,944 / 191,206 = 6.2%, Response rate (143) = 8,477 / 155,946 = 5.4%,
Response rate (142) = 1,707 / 37,453 = 4.5%, Response rate (141) = 282 / 9,006 = 3.1%,
Response rate (135) = 4,973 / 157,926 = 3.1%, Response rate (134) = 6,348 / 181,741 = 3.4%,
Response rate (133) = 3,540 / 125,019 = 2.8%, Response rate (132) = 392 / 21,624 = 1.8%,
Response rate (131) = 34 / 5,262 = 0.6%, Response rate (125) = 1,458 / 127,525 = 1.1%,
Response rate (124) = 2,185 / 166,505 = 1.3%, Response rate (123) = 1,272 / 111,988 = 1.1%,
Response rate (122) = 50 / 13,853 = 0.3%, Response rate (121) = 2 / 2025 = 0.09%,
Response rate (115) = 175 / 108,691 = 0.1%, Response rate (114) = 61 / 59,179 = 0.1%,
Response rate (113) = 161 / 93,519 = 0.1%, Response rate (112) = 1 / 12029 = 0,0083,
Response rate (111) = 0 / 3,599 = 0%

Now for our next subquestion, how many to mail based on an estimate of ROI of the campaign, we could calculate the ROI of every segment, and decide to mail on the ones that have a positive ROI. The ROI would be calculated by multiplying the average profit of $30 per purchase with the number of customers and deducting the normalized cost of $1 per mailing multiplied by the customers for every segment.

However, since this is again a time-consuming approach we can think of another one, that is also equaly accurate. We can decide on which to mail, by an estimate and not a calculation of ROI, and this is also implied by the formulation of the question.

We know from the lectures that we can ran a campaign by determining a break-even point. In other words, by determining how many respondents must purchases in order to generate adequate profit to pay for the cost of our campaign. After this point, we can generate a positive ROI.

As a result, the break even response rate is calculated through ROI, since it is equal to the normalized cost per mailing (cost per piece) divided by the average profit per purchase (revenue per sale). Hence, in our case the break even response rate is equal to 1/30. Hence, we make the decision to mail the segments that have a response rate that is greater than the break even response rate. In other words, mail the segments that have a response rate greater than 1/30 = 0.0034 or greater than 3.4%.

Based on our previous calculation, we will mail the following segments:

“555”, “554”, “553”, “552”, “551”, “545”, “544”, “543”, “542”, “541”, “535”, “534”, “533”, “532”, “531”, “525”, “524”, “523”, “522”, “515”, “455”, “454”, “453”, “452”, “451”, “445”, “444”, “443”, “442”, “441”, “435”, “434”, “433”, “432”, “425”, “424”, “423”, “355”, “354”, “353”, “352”, “351”, “345”, “344”, “343”, “342”, “341”, “335”, “334”, “333”, “255”, “254”, “253”, “252”, “251”, “245”, “244”, “243”, “242”, “235”, “234”, “155”, “154”, “153”, “152”, “151”, “145”, “144”, “143”, “142”.

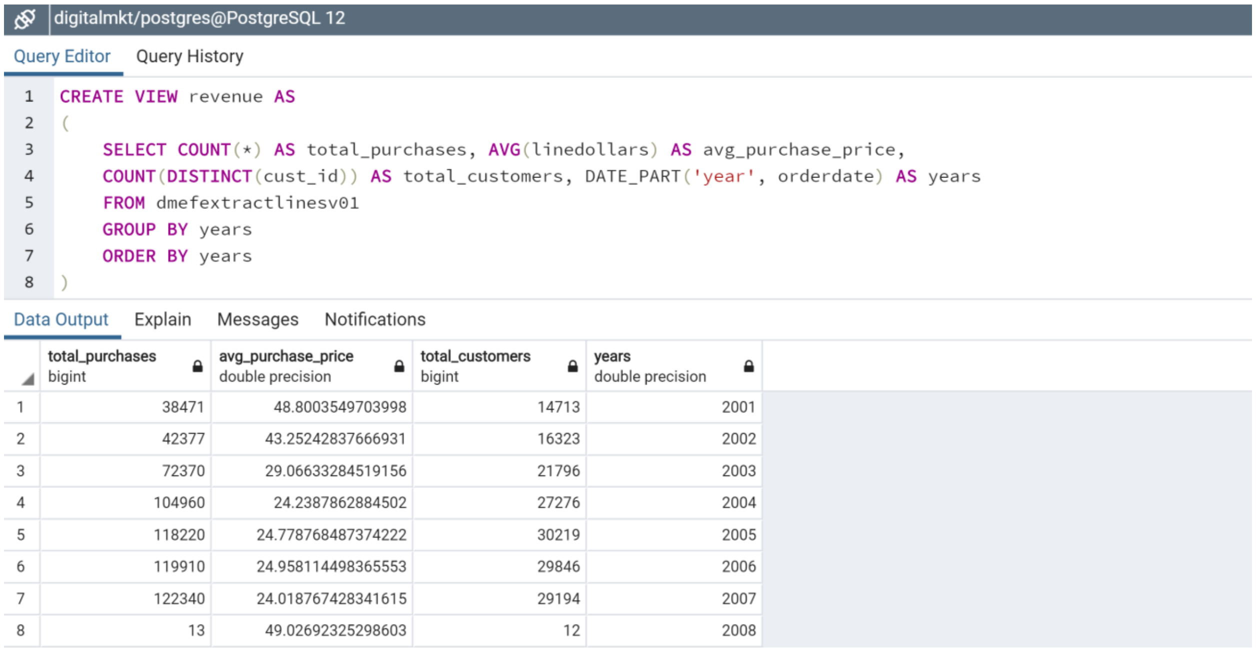
Therefore, we recommend to mail 70 out of the total 125 segments, in other words mail 56% of the segments. If the percentage of targeting seems to large for the company, we can decrease the time window, of the order to be placed after the contact, from 14 days to something smaller.

QUESTION 3

A.

First, in order to calculate the average CLV of a customer, we have to recall to its formula. We start by finding the expected revenue (margin) from this customer in period t.

In other words, we multiple the number of purchases with the average purchase price and then divide that number with the total number of customers for the year. The “Mt” is calculated in the following query. Note that this time we did not round the “linedollars” field, since we do not want to affect the final average CLV value.

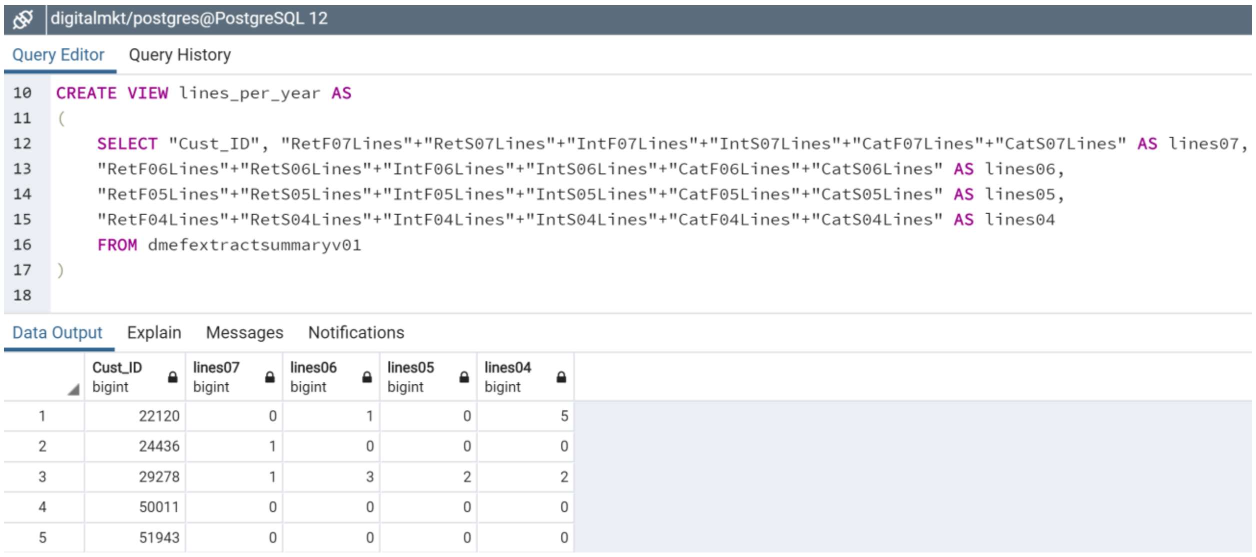

digitalmkt/postgres@PostgreSQL 12

```
CREATE VIEW revenue AS
(
    SELECT COUNT(*) AS total_purchases, AVG(linedollars) AS avg_purchase_price,
    COUNT(DISTINCT(cust_id)) AS total_customers, DATE_PART('year', orderdate) AS years
    FROM dmefextractlinesv01
    GROUP BY years
    ORDER BY years
)
```

	total_purchases bigint	avg_purchase_price double precision	total_customers bigint	years double precision
1	38471	48.8003549703998	14713	2001
2	42377	43.25242837666931	16323	2002
3	72370	29.06633284519156	21796	2003
4	104960	24.2387862884502	27276	2004
5	118220	24.778768487374222	30219	2005
6	119910	24.958114498365553	29846	2006
7	122340	24.018767428341615	29194	2007
8	13	49.02692325298603	12	2008

Screenshot 13: SQL Query and output for question 3A – Mt.

After that we will calculate the retention rate. Recall from lectures that the retention rate is equal to the fraction of customers who continue their relationship with the firm in a given period. As a result, in order to calculate it we need the total number of customers for two successive years. We will extract that information with the following query:


digitalmkt/postgres@PostgreSQL 12

```
CREATE VIEW lines_per_year AS
(
    SELECT "Cust_ID", "RetF07Lines"+"RetS07Lines"+"IntF07Lines"+"IntS07Lines"+"CatF07Lines"+"CatS07Lines" AS lines07,
    "RetF06Lines"+"RetS06Lines"+"IntF06Lines"+"IntS06Lines"+"CatF06Lines"+"CatS06Lines" AS lines06,
    "RetF05Lines"+"RetS05Lines"+"IntF05Lines"+"IntS05Lines"+"CatF05Lines"+"CatS05Lines" AS lines05,
    "RetF04Lines"+"RetS04Lines"+"IntF04Lines"+"IntS04Lines"+"CatF04Lines"+"CatS04Lines" AS lines04
    FROM dmefextractsummaryv01
)
```

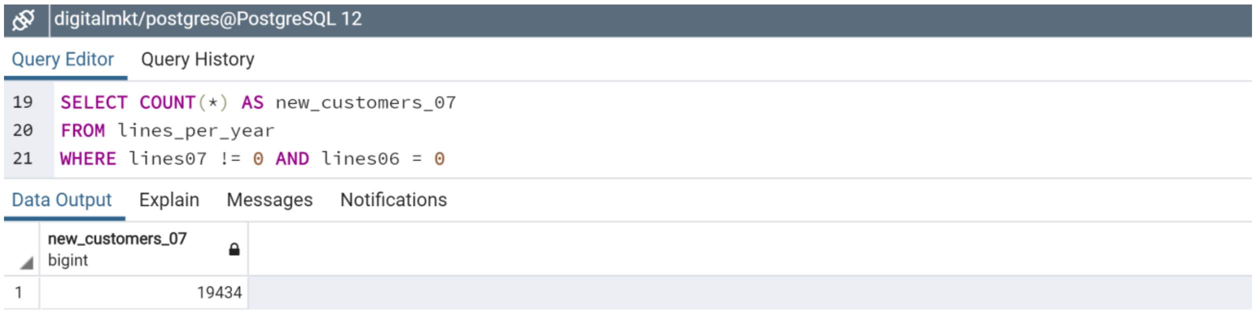
	Cust_ID bigint	lines07 bigint	lines06 bigint	lines05 bigint	lines04 bigint
1	22120	0	1	0	5
2	24436	1	0	0	0
3	29278	1	3	2	2
4	50011	0	0	0	0
5	51943	0	0	0	0

Screenshot 14: SQL Query and output for question 3A – Rt.

Now we can find the total new customers for years 2007, 2006 and 2005. Note that calculating total new customers for 2004 is more difficult, since we do not have detailed data for 2003.

Based on our second to last query ("Screenshot 12"), we can now calculate the retention rates for the three previously mentioned years.

Previously we calculate that the total customers for 2007 and 2006 were 29,194 and 29,846 respectively. Now we calculate the new customers from 2006 to 2007 as follows:


digitalmkt/postgres@PostgreSQL 12

Query Editor Query History

```
SELECT COUNT(*) AS new_customers_07
FROM lines_per_year
WHERE lines07 != 0 AND lines06 = 0
```

Data Output Explain Messages Notifications

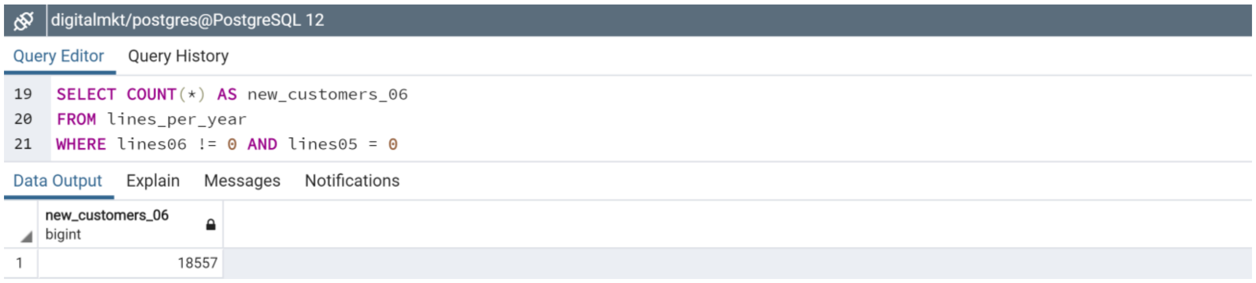
```
   new_customers_07
   bigint
1             19434
```

Screenshot 15: SQL Query and output question 3A – Rt (2007).

Therefore, for 2007 we have: retention rate = (29,194 - 19,434) / 29,846 = 32.7%.

Now we follow the same procedure for the rest two years that we have enough data to calculate the retention rate.

Previously we calculate that the total customers for 2006 and 2005 were 29,846 and 30,219 respectively. Now we calculate the new customers from 2005 to 2006 as follows:


digitalmkt/postgres@PostgreSQL 12

Query Editor Query History

```
SELECT COUNT(*) AS new_customers_06
FROM lines_per_year
WHERE lines06 != 0 AND lines05 = 0
```

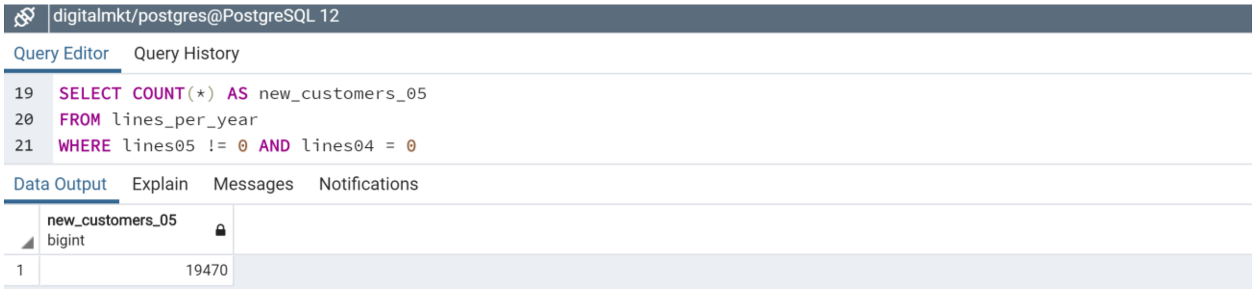
Data Output Explain Messages Notifications

```
   new_customers_06
   bigint
1             18557
```

Screenshot 16: SQL Query and output for question 3A – Rt (2006).

Therefore, for 2006 we have: retention rate = (29,846 - 18,557) / 30,219 = 37.3%.

Previously we calculate that the total customers for 2005 and 2004 were 27,276 and 30,219 respectively. Now we calculate the new customers from 2004 to 2005 as follows:


digitalmkt/postgres@PostgreSQL 12

Query Editor Query History

```
SELECT COUNT(*) AS new_customers_05
FROM lines_per_year
WHERE lines05 != 0 AND lines04 = 0
```

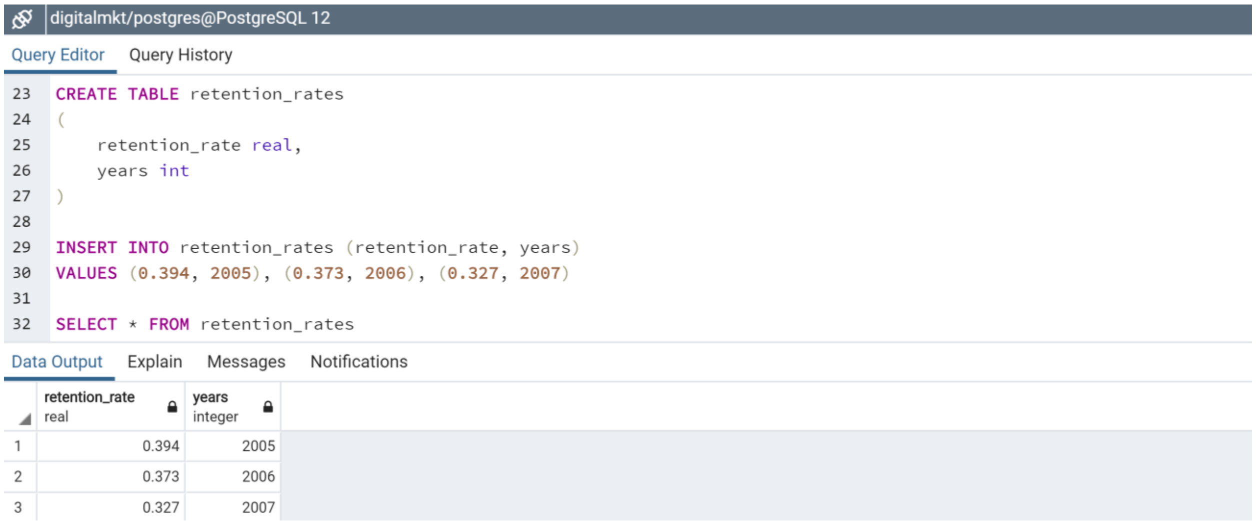
Data Output Explain Messages Notifications

```
   new_customers_05
   bigint
1             19470
```

Screenshot 17: SQL Query and output for question 3A – Rt (2005).

Therefore, for 2005 we have: retention rate = (30,219 - 19,470) / 27,276 = 39.4%.

Now that we calculated the retention rates, we store them in a table to use them later and determine the average CLV of a customer, as follows:



digitalmkt/postgres@PostgreSQL 12

Query Editor | Query History

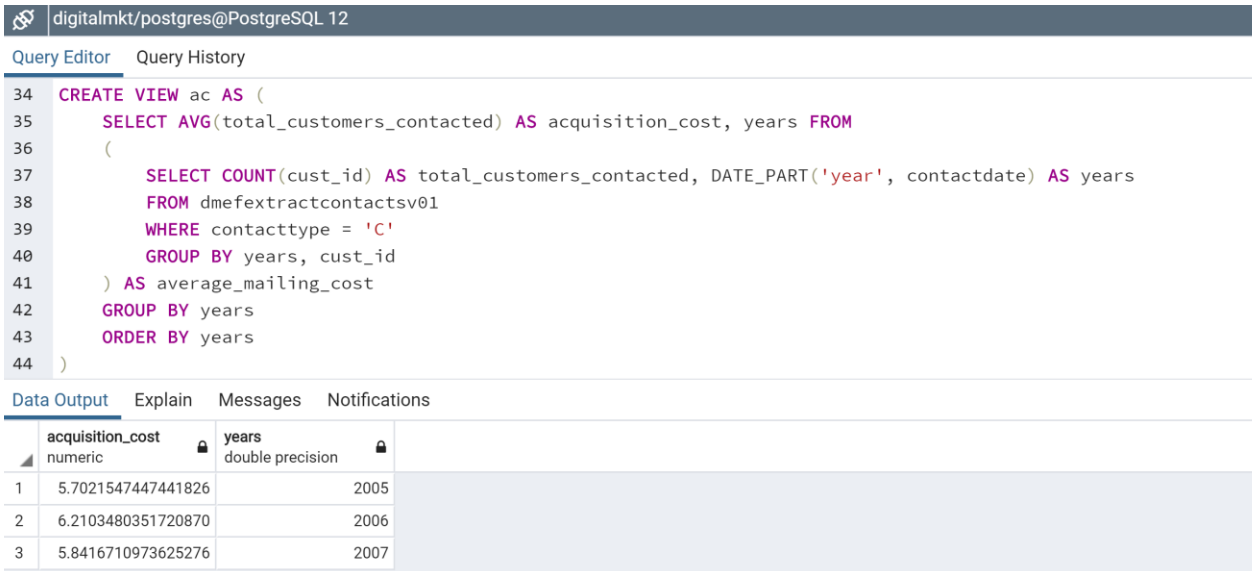
```sql
CREATE TABLE retention_rates
(
    retention_rate real,
    years int
)

INSERT INTO retention_rates (retention_rate, years)
VALUES (0.394, 2005), (0.373, 2006), (0.327, 2007)

SELECT * FROM retention_rates
```

Data Output | Explain | Messages | Notifications

	retention_rate real	years integer
1	0.394	2005
2	0.373	2006
3	0.327	2007

Screenshot 18: SQL Query and output for question 3A – AC (Acquisition Cost).

Now we calculate the acquisition cost (AC) as follows:



digitalmkt/postgres@PostgreSQL 12

Query Editor | Query History

```sql
CREATE VIEW ac AS (
    SELECT AVG(total_customers_contacted) AS acquisition_cost, years FROM
    (
        SELECT COUNT(cust_id) AS total_customers_contacted, DATE_PART('year', contactdate) AS years
        FROM dmefextractcontactsv01
        WHERE contacttype = 'C'
        GROUP BY years, cust_id
    ) AS average_mailing_cost
    GROUP BY years
    ORDER BY years
)
```

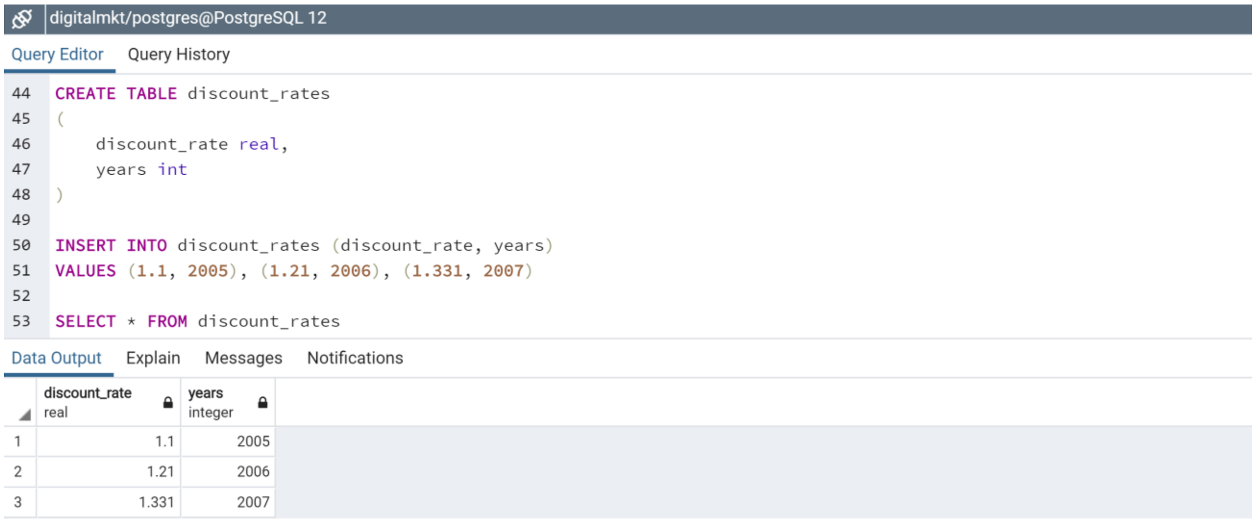
Data Output | Explain | Messages | Notifications

	acquisition_cost numeric	years double precision
1	5.7021547447441826	2005
2	6.2103480351720870	2006
3	5.8416710973625276	2007

Screenshot 19: SQL Query and output for question 3A – acquisition cost.

Note that we included only the customers that were contacted through catalog mailng.

After that, we calculate the discount rate for the future revenue. We choose a discount rate of 10% as we did in the lectures. Hence, the first year will have a discount rate of 1 + 0.1 x 1 = 1.1 the second year a discount rate of 1.1 + 0.1 x 1.1 = 1.21 and the third one of 1.21 + 0.1 x 1.21 = 1.331. We calculate the discount rate for each of the three years and store it in a table as follows:


digitalmkt/postgres@PostgreSQL 12

```
CREATE TABLE discount_rates
(
    discount_rate real,
    years int
)

INSERT INTO discount_rates (discount_rate, years)
VALUES (1.1, 2005), (1.21, 2006), (1.331, 2007)

SELECT * FROM discount_rates
```

	discount_rate real	years integer
1	1.1	2005
2	1.21	2006
3	1.331	2007

Screenshot 20: SQL Query and output for question 3A – i (discount rates table).

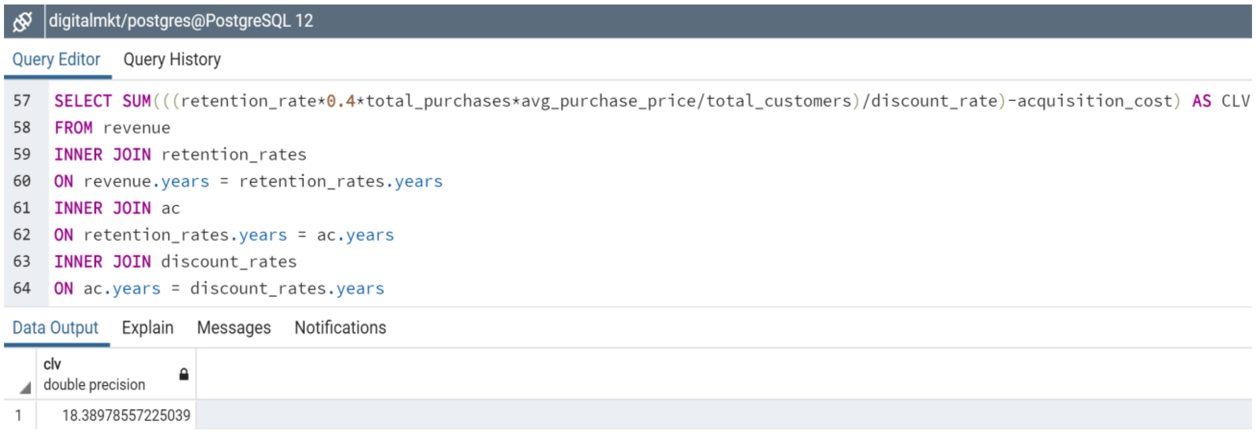
Finally, we can calculate the average CLV of a customer. First, recall to the CLV formula:

$$CLV = \sum \frac{retention\ rate\ *\ revenue}{discount\ rate} - acquisition\ cost$$

Or in other words:

$$CLV = \sum \frac{retention\ rate\ *\ (\frac{total\,purchases\,*\,avg\,purchase\,price}{total\,customers})}{discount\ rate} - acquisition\ cost$$

Note that in the two above formulas, when we mention "revenue" and "total purchases x average purchase price / total customers", we mean the expected revenue margin. We assume that the the margin per purchase for the given firm is equal to 40% as this is close to the average margin for a retail business. We now join "revenue", "retention_rates", "acquisition_cost" and "discount_rates" table/view as follows:


digitalmkt/postgres@PostgreSQL 12

```
SELECT SUM(((retention_rate*0.4*total_purchases*avg_purchase_price/total_customers)/discount_rate)-acquisition_cost) AS CLV
FROM revenue
INNER JOIN retention_rates
ON revenue.years = retention_rates.years
INNER JOIN ac
ON retention_rates.years = ac.years
INNER JOIN discount_rates
ON ac.years = discount_rates.years
```

	clv double precision
1	18.38978557225039

Screenshot 21: SQL Query and output for question 3A – CLV calculation

Therefore, the average CLV of a customer is $18.3.

B.

Management believes different types of customers have different CLVs. Indeed, each customer that belongs to a different segment, has a different lifetime value. We will now focus on how we would do such a project, so the results are useful for the business based on the data that we have.

What we can say, is that for every different segment we will have a different CLV. Recall to our example in lectures, where we divided for lower and greater than $50. Moreover, each customer has different characteristics regarding demographics or behaviour and thus different CLV. Hence, every type of customer, will have a different retention rate, revenue and acquisition cost, since all that change. In order to calculate them, we would focus on the characteristics of every RFM segment.

More specifically, we could find the different customer segments by running a linear regression model. Thus, we could calculate the mean retention rate for every customer. Similarly, we could run the linear regression to find the mean acquisition cost and the mean number of purchases, and as a result the mean revenue. We can also run another linear regression to test how customers perform on different marketing channels, as we did in this assignment for catalog mailing. Knowing how each segment of customers performs on the different marketing channels will enable us maximise the ROI.

Conclusion

In the coursework assignment we analysed a big data set from a multichannel company with sales of several hundred million dollars per year and produced digital marketing insights. The insights involved demographic targeting, RFM segmentation, estimation of ROI and CLV calculation. For data analysis mainly SQL as well as Python were used.