Mental Health Clinic

YECHEZKEL BOTWINICK , YAAKOV FREDMAN

Database project
Instructor – Aryeh Vizen

את הדו"ח כתבנו בשפה האנגלית למרות הקושי להתבטא בשפה שאינה שפת אם, מתוך רצון להבין את עצמנו טוב יותר ל"עולם האמיתי" של ההייטק שבו השפה המקובלת היא אנגלית.

Mental Health Clinic

The system is designed to describe a database for a mental health clinic. The clinic offers mental health treatments for all those seeking help. The services include traditional therapy, cognitive behavioral therapy, and various treatment methods.
We have a variety of specialists on our staff, all highly trained professionals.

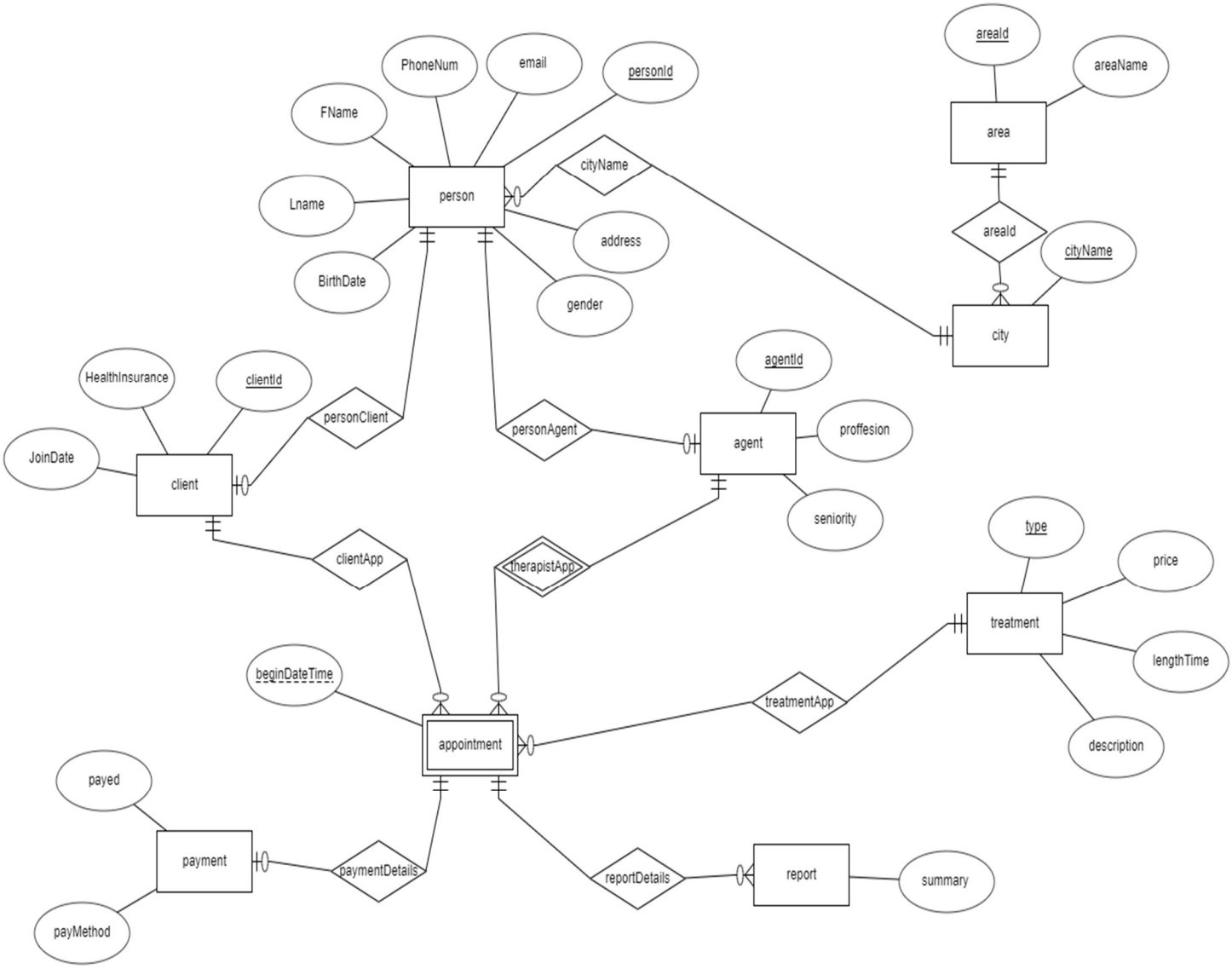
In order to give the best service to the clients there is a need for a database for operating the clinic.

The database will manage all the details for the therapists, clients, payments and scheduling appointments.

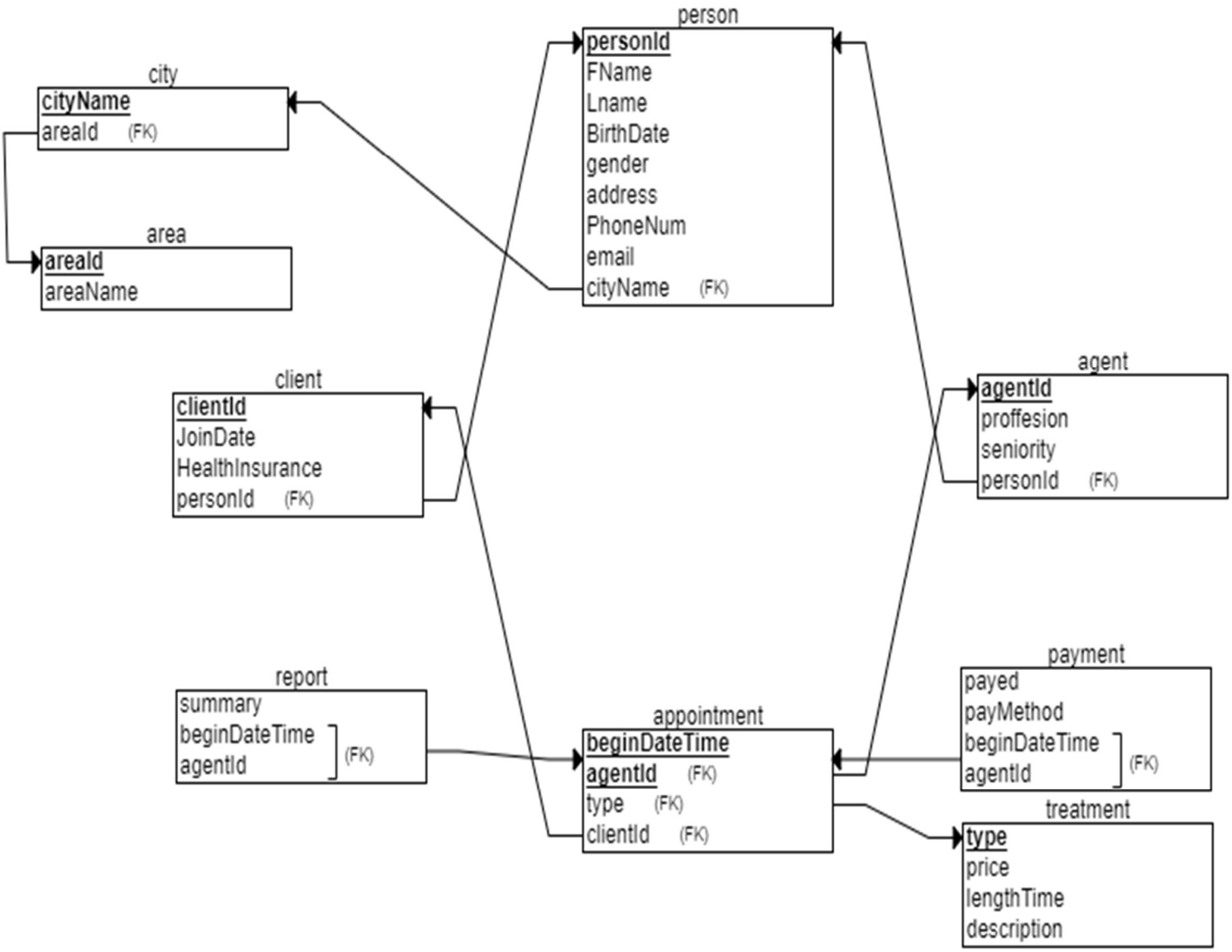
This document will describe the system.

Part 1

Entity Relationship Diagram (ERD)



Data structure diagram (DSD)



Entity details

All entities are not weak entities unless written otherwise. **(**only appointment is a weak entity).

area - area in country - for example: in Israel there are 4: Jerusalem, North, South and center

city - a list of cites
areaid - foreign key of area

treatment - type of a treatment, including the length time and the price for specific treatment

person - basic details about client/agent (agent = therapist).
cityName - foreign key to city

client - the patient
personClient - foreign key to person

agent - the therapist
personAgent - foreign key to person

appointment - all appointments, the core of the system
the only weak entity in our system.
the primary key is: beginDateTime and agentId because the agent can have only one appointment at the same time.
clientApp - foreign key to client
therapistApp - foreign key to agent
treatmentApp - foreign key to treatment
reportDetails - foreign key to report
paymentDetails - foreign key to payment

payment - tells us whether the client payed for specific appointment and the payment methods
paymentDetails - foreign key to appointment

report - optional, gives us more details about appointment (summary)

All tables are in 3nf since the definition of a table is in a third normal form if it is a second normal form and there is no functional dependency between two components that are not part of the primary key. In other words, any field other than a key field must not be dependent on a field other than a key field. In our tables there isn't any dependency from a non-key to any other field. Therefore, it is at least in 3nf.

Sql scripts

Create Tables: From erdplus (Export Sql Beta)

```
CREATE TABLE treatment
(
  type VARCHAR  NOT NULL,
  price NUMERIC NOT NULL,
  lengthTime INT NOT NULL,
  description VARCHAR NOT NULL,
  PRIMARY KEY (type)
);

CREATE TABLE area
(
  areaId INT NOT NULL,
  areaName VARCHAR(25) NOT NULL,
  PRIMARY KEY (areaId)
);

CREATE TABLE city
(
  cityName VARCHAR(20) NOT NULL,
  areaId INT NOT NULL,
  PRIMARY KEY (cityName),
  FOREIGN KEY (areaId) REFERENCES area(areaId)
);

CREATE TABLE person
(
  FName VARCHAR(20) NOT NULL,
  Lname VARCHAR(20) NOT NULL,
  BirthDate DATE NOT NULL,
  gender CHAR NOT NULL,
  address VARCHAR NOT NULL,
  PhoneNum INT NOT NULL,
  email_ VARCHAR NOT NULL,
  personId INT NOT NULL,
  cityName VARCHAR(20) NOT NULL,
  PRIMARY KEY (personId),
  FOREIGN KEY (cityName) REFERENCES city(cityName)
);

CREATE TABLE client
(
  JoinDate DATE NOT NULL,
  HealthInsurance VARCHAR NOT NULL,
```

```
  clientId INT NOT NULL,
  personId INT NOT NULL,
  PRIMARY KEY (clientId),
  FOREIGN KEY (personId) REFERENCES person(personId)
);

CREATE TABLE agent
(
  proffesion VARCHAR NOT NULL,
  seniority INT NOT NULL,
  agentId INT NOT NULL,
  personId INT NOT NULL,
  PRIMARY KEY (agentId),
  FOREIGN KEY (personId) REFERENCES person(personId)
);

CREATE TABLE appointment
(
  beginDateTime DATE NOT NULL,
  agentId INT NOT NULL,
  type VARCHAR NOT NULL,
  clientId INT NOT NULL,
  PRIMARY KEY (beginDateTime, agentId),
  FOREIGN KEY (agentId) REFERENCES agent(agentId),
  FOREIGN KEY (type) REFERENCES treatment(type),
  FOREIGN KEY (clientId) REFERENCES client(clientId)
);

CREATE TABLE report
(
  summary_ VARCHAR NOT NULL,
  beginDateTime DATE NOT NULL,
  agentId INT NOT NULL,
  FOREIGN KEY (beginDateTime, agentId) REFERENCES appointment(beginDateTime, agentId)
);

CREATE TABLE payment
(
  payed  NOT NULL,
  payMethod VARCHAR NOT NULL,sn
  beginDateTime DATE NOT NULL,
  agentId INT NOT NULL,
  FOREIGN KEY (beginDateTime, agentId) REFERENCES appointment(beginDateTime, agentId)
);
```

Drop tables

```
drop table city;
drop table area;
drop table report;
drop table payment ;
drop table treatment;
drop table appointment;
drop table client;
drop table therapist;
drop table person;
```

Inserting data into tables

manually:

insert into area (AREAID, AREANAME)
values (1, 'north);

	AREAID	AREANAME
1	1	north

Data generator

To city table:
We wanted to make sure that only areaid's that exist in the system will be chosen therefor we used List(select areaid from area) since it is a foreign key.

CITY

Owner: SYSTEM | Table: CITY | Number of records: 10..20

Name	Type	Size	Data	Master
CITYNAME	VARCHAR2	20	City	
AREAID	NUMBER		List(select areaid from area)	

Result:

SQL Output Statistics

```
SELECT * FROM city;
```

	CITYNAME	AREAID
1	Monument	1
2	Mogliano Veneto	2
3	New boston	2
4	Barcelona	2
5	Holts Summit	2
6	Woking	3
7	Regensburg	3
8	Oldenburg	4
9	Baltimore	4
10	Sao roque	4
11	Dresden	4
12	Bad Camberg	4

To person table:

createTable.sql | pesrson.gd | Data Generator

PERSON

Owner: SYSTEM | Table: PERSON | Number of records: 10..20

Name	Type	Size	Data	Master
FNAME	VARCHAR2	20	FirstName	
LNAME	VARCHAR2	20	LastName	
BIRTHDATE	DATE		random(01/01/1965,31/12/2003)	
GENDER	CHAR	1	List('F', 'M')	
ADDRESS	VARCHAR2	50	Address1	
PHONENUM	NUMBER		Random(500000000, 560000000)	
EMAIL_	VARCHAR2	30	Email	
PERSONID	NUMBER		Random(10000000, 100000000)	
CITYNAME	VARCHAR2	20	List(select cityname from city)	

Result:

SQL | Output | Statistics

```
SELECT * FROM person;
```

	FNAME	LNAME	BIRTHDATE	GENDER	ADDRESS	PHONENUM	EMAIL_	PERSONID	CITYNAME
1	Wang	Mifune	25-Dec-95	F	39 Or-yehuda Road	508447964	wang.mifune@eastmankodak.de	44573957	Holts Summ
2	Grace	Viterelli	19-Feb-77	M	36 Mazar Drive	550107496	grace.viterelli@vspan.uk	26338715	Baltimore
3	Sal	Wopat	24-Jan-88	M	76 Porter Blvd	505303612	salw@nha.com	55441950	Oldenburg
4	Fairuza	Fiennes	14-Mar-79	F	26 Austin Street	503826638	fairuza.fiennes@kingland.ca	32442690	Regensburg
5	Mike	Collins	19-Jun-85	M	1 Pottendorf Blvd	541057028	mike.c@taycorfinancial.br	79016729	Sao roque
6	Anna	Gaines	11-Feb-96	F	85 Ani Street	515451579	anna.gaines@bioreliance.si	27934537	Monument
7	Milla	Phoenix	08-Dec-72	F	27 Chloe Street	555975629	milla.p@bigdoughcom.at	13297832	Mogliano Ve
8	Lorraine	Thompson	28-May-71	M	77 Haslam Street	513216219	lorraine.thompson@hewlettpacka	78614740	Barcelona
9	Kate	Conlee	11-Feb-02	M	16 Kenneth Street	522215558	kate.conlee@otbd.uk	64789966	Regensburg
10	Merrill	Henriksen	08-Aug-65	F	50 Lorenz Drive	532128083	merrill.henriksen@keith.com	91412598	Woking

To client table:

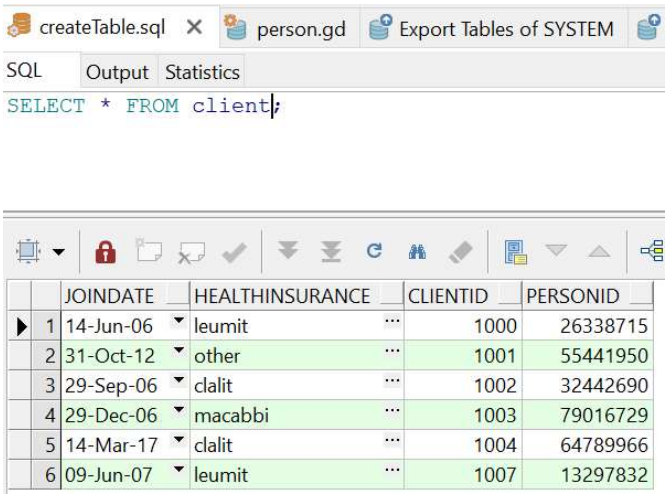
createTable.sql | pesrson.gd | clientDataGen.gd

CLIENT

Owner: SYSTEM | Table: CLIENT | Number of records: 10..20

Name	Type	Size	Data	Master
JOINDATE	DATE		Random(01/01/2005,31/12/2019)	
HEALTHINSURANCE	VARCHAR2	20	List('macabbi', 'leumit','meuhedet','clalit','other')	
CLIENTID	NUMBER		Sequence(1000, [1])	
PERSONID	NUMBER		List(select distinct personid from person)	

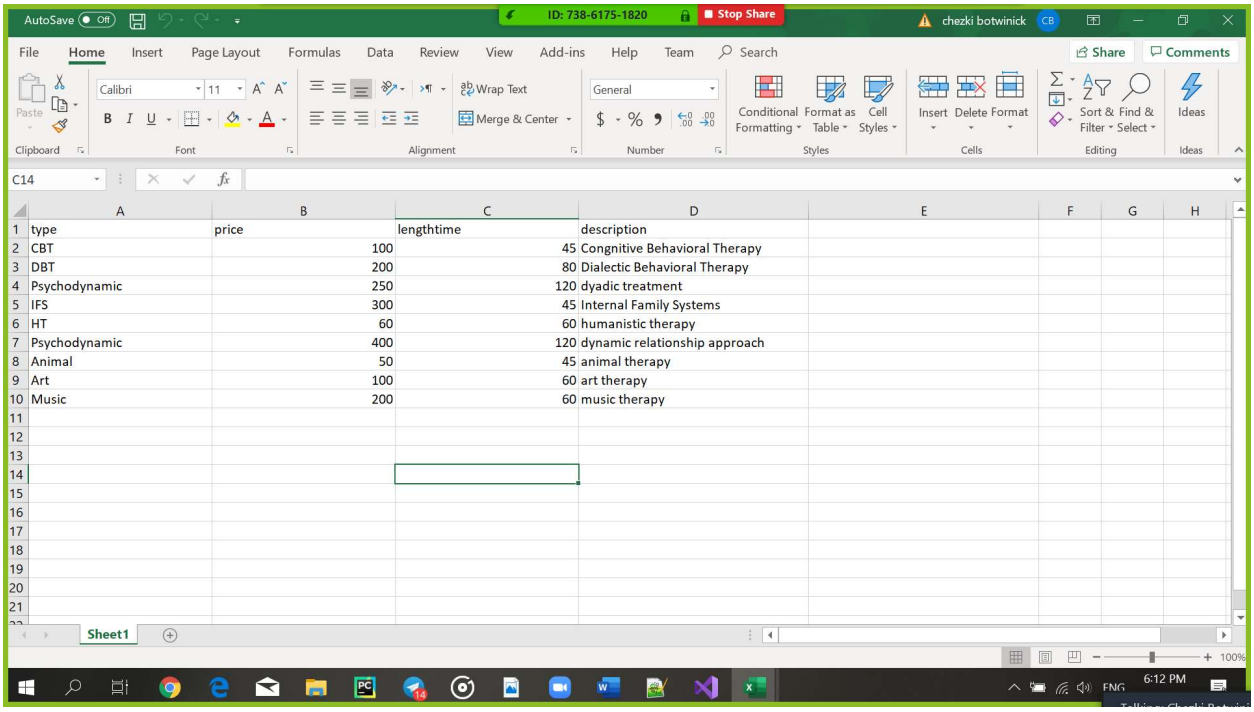
Result:



	JOINDATE	HEALTHINSURANCE	CLIENTID	PERSONID
1	14-Jun-06	leumit	1000	26338715
2	31-Oct-12	other	1001	55441950
3	29-Sep-06	clalit	1002	32442690
4	29-Dec-06	macabbi	1003	79016729
5	14-Mar-17	clalit	1004	64789966
6	09-Jun-07	leumit	1007	13297832

Using Text importer

For treatment from .csv file:



The definitions for the text import are within the file attached.

After:

```
SQL  Output  Statistics
select * from treatment
```

	TYPE	PRICE	LENGTHTIME	DESCRIPTION
1	CBT	100	45	Congnitive Behavioral Therapy
2	DBT	200	80	Dialectic Behavioral Therapy
3	Psychodynamic	250	120	dyadic treatment
4	IFS	300	45	Internal Family Systems
5	HT	60	60	humanistic therapy
6	Animal	50	45	animal therapy
7	Art	100	60	art therapy
8	Music	200	60	music therapy

For agent:

The file contains the following:

proffesion

A	B	C	D	E	F	G	H
proffesion	seniority	agentId	personId				
therapist	3	1000	13297832				
therapist	6	1001	32442690				
psycholog	2	1002	79016729				
psycholog	8	1003	91412598				

After:

```
SQL  Output  Statistics
select * from agent
```

	PROFFESION	SENIORITY	AGENTID	PERSONID
1	therapist	3	1000	13297832
2	therapist	6	1001	32442690
3	psycholog	2	1002	79016729
4	psycholog	8	1003	91412598

Into appointment:

appointment - Notepad

File Edit Format View Help

```
insert into SYSTEM.APPOINTMENT (BEGINDATETIME, AGENTID, TYPE, CLIENTID)
values (to_date('08-01-2020 14:15:00', 'dd-mm-yyyy hh24:mi:ss'), 1003, 'CBT', 1000);

insert into SYSTEM.APPOINTMENT (BEGINDATETIME, AGENTID, TYPE, CLIENTID)
values (to_date('10-01-2020 11:30:00', 'dd-mm-yyyy hh24:mi:ss'), 1000, 'CBT', 1002);

insert into SYSTEM.APPOINTMENT (BEGINDATETIME, AGENTID, TYPE, CLIENTID)
values (to_date('13-01-2020 15:00:00', 'dd-mm-yyyy hh24:mi:ss'), 1001, 'Animal', 1000);

insert into SYSTEM.APPOINTMENT (BEGINDATETIME, AGENTID, TYPE, CLIENTID)
values (to_date('15-02-2020 18:15:00', 'dd-mm-yyyy hh24:mi:ss'), 1003, 'DBT', 1000);

insert into SYSTEM.APPOINTMENT (BEGINDATETIME, AGENTID, TYPE, CLIENTID)
values (to_date('18-01-2020 02:30:00', 'dd-mm-yyyy hh24:mi:ss'), 1001, 'Psychodynamic', 1003);

insert into SYSTEM.APPOINTMENT (BEGINDATETIME, AGENTID, TYPE, CLIENTID)
values (to_date('28-02-2020 06:45:00', 'dd-mm-yyyy hh24:mi:ss'), 1001, 'HT', 1002);
```

Ln 6, Col 1 | 100% | Windows (CRLF) | UTF-8

The result:

SQL | Output | Statistics

```
select * from appointment
```

	BEGINDATETIME	AGENTID	TYPE	CLIENTID
1	10-Jan-20 11:30:00 AM	1000	CBT	1002
2	13-Jan-20 3:00:00 PM	1001	Animal	1000
3	15-Feb-20 6:15:00 PM	1003	DBT	1000
4	28-Feb-20 6:45:00 AM	1001	HT	1002
5	15-Jan-20 1:30:00 PM	1000	CBT	1007
6	03-Jan-20 8:15:00 PM	1000	Art	1003
7	10-Feb-20 4:30:00 AM	1002	HT	1003
8	04-Jan-20 5:15:00 PM	1002	HT	1000
9	30-Jan-20 6:30:00 PM	1002	CBT	1002
10	15-Jan-20 1:15:00 PM	1001	DBT	1007
11	19-Jan-20 5:45:00 PM	1003	Art	1007
12	13-Feb-20 11:15:00 PM	1000	CBT	1001
13	08-Feb-20 11:30:00 PM	1003	HT	1003
14	09-Feb-20 2:00:00 PM	1000	CBT	1001

Into payment:

From .cvs File

payed	paymethod	BEGINDATETIME	AGENTID
0	no method	01-10-20 11:30	1000
1	credit card	1/13/2020 15:00	1001
1	check	2/15/2020 18:15	1003
1	cash	2/28/2020 6:45	1001
1	credit card	1/15/2020 13:30	1000
0	no method	01-03-20 20:15	1000
1	credit card	02-10-20 4:30	1002
1	credit card	01-04-20 17:15	1002
1	check	1/30/2020 18:30	1002
0	no method	1/15/2020 13:15	1001
1	cash	1/19/2020 17:45	1003
1	cash	2/13/2020 23:15	1000
0	credit card	02-08-20 23:30	1003

After:

SQL Output Statistics

```
select * from payment
```

	PAYED	PAYMETHOD	BEGINDATETIME	AGENTID
1	1	credit card	13-Jan-20 3:00:00 PM	1001
2	1	check	15-Feb-20 6:15:00 PM	1003
3	1	cash	28-Feb-20 6:45:00 AM	1001
4	1	credit card	15-Jan-20 1:30:00 PM	1000
5	1	credit card	10-Feb-20 4:30:00 AM	1002
6	1	credit card	04-Jan-20 5:15:00 PM	1002
7	1	check	30-Jan-20 6:30:00 PM	1002
8	1	cash	19-Jan-20 5:45:00 PM	1003
9	1	cash	13-Feb-20 11:15:00 PM	1000
10	1	credit card	09-Feb-20 2:00:00 PM	1000
11	0	no method	10-Jan-20 11:30:00 AM	1000
12	1	credit card	13-Jan-20 3:00:00 PM	1001
13	1	check	15-Feb-20 6:15:00 PM	1003
14	1	cash	28-Feb-20 6:45:00 AM	1001

Into reports:

(we added short summary for every session where obviously it could be longer, as in reality.)

From the .cvs file as follows:

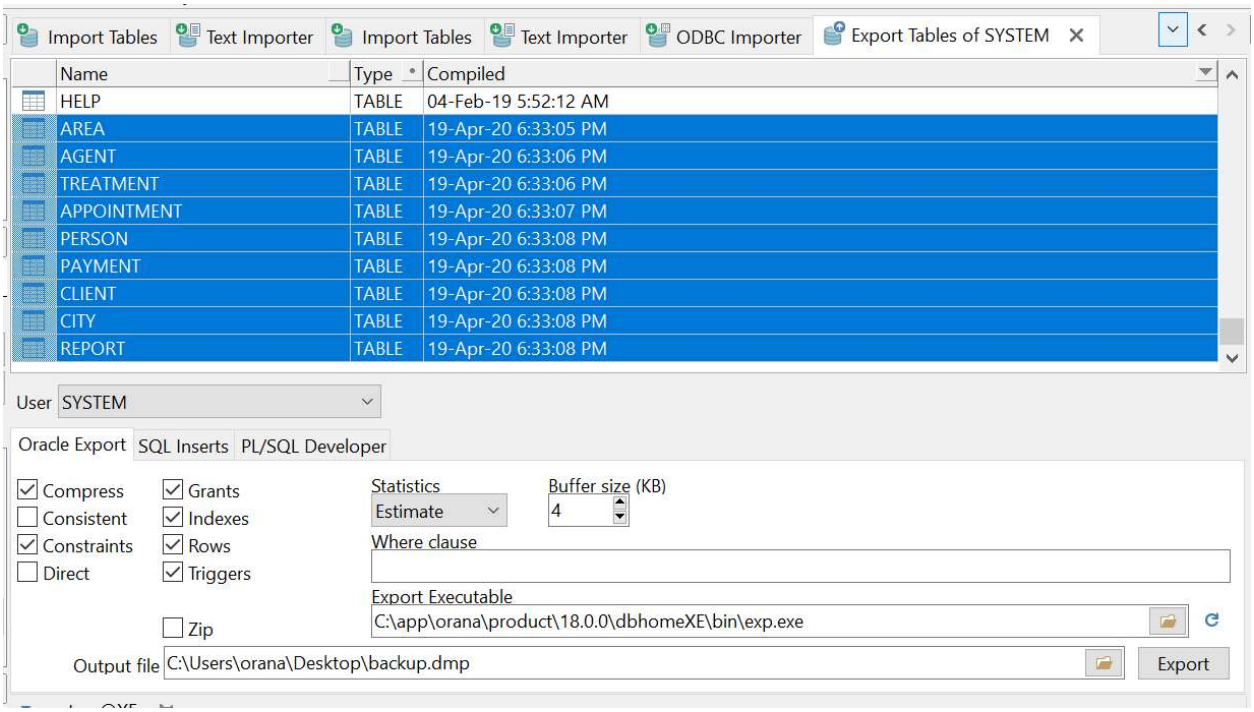
C4 | 1003

	A	B	C	D
1	summary	BEGINDATETIME	AGENTID	
2	ok	01-10-20 11:30	1000	
3	need one	1/13/2020 03:00:00 PM	1001	
4	ok	2/15/2020 06:15:00 PM	1003	
5	there is nc	2/28/2020 06:45:00 AM	1001	
6	fine	1/15/2020 01:30:00 PM	1000	
7	need mor	01-03-20 20:15	1000	
8	ok	02-10-20 4:30	1002	
9	need one	01-04-20 17:15	1002	
10	ok	1/30/2020 06:30:00 PM	1002	
11	there is nc	1/15/2020 01:15:00 PM	1001	
12	fine	1/19/2020 05:45:00 PM	1003	
13	need mor	2/13/2020 11:15:00 PM	1000	
14	finished	02-08-20 23:30	1003	
15	all right	02-09-20 14:00	1000	
16	date with	01-05-20 16:00	1000	
17	no details	01-08-20 14:15	1003	
18				

After:

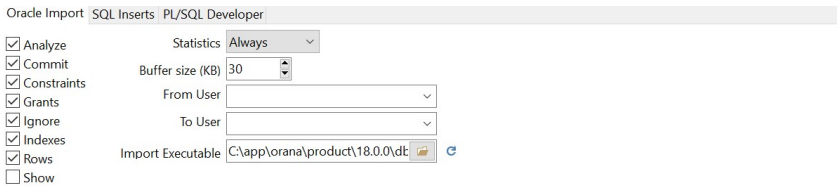
	SUMMARY_	BEGINDATETIME	AGENTID
1	ok	10-Jan-20 11:30:00 AM	1000
2	need one more date	13-Jan-20 3:00:00 PM	1001
3	ok	15-Feb-20 6:15:00 PM	1003
4	there is no progress	28-Feb-20 6:45:00 AM	1001
5	fine	15-Jan-20 1:30:00 PM	1000
6	need more help	03-Jan-20 8:15:00 PM	1000
7	ok	10-Feb-20 4:30:00 AM	1002
8	need one more date	04-Jan-20 5:15:00 PM	1002
9	ok	30-Jan-20 6:30:00 PM	1002
10	there is no progress	15-Jan-20 1:15:00 PM	1001
11	fine	19-Jan-20 5:45:00 PM	1003
12	need more help	13-Feb-20 11:15:00 PM	1000
13	finished	08-Feb-20 11:30:00 PM	1003
14	all right	09-Feb-20 2:00:00 PM	1000

Back up using export tables:



This will create a file named backup.dmp and with that we will be able to restore the tables.

In order to restore the backup, we do the following:



Part 2

Queries

1. Returns the agent that is the most active – has the most appointments. Value occurrence is the number of appointments the therapist has.

```
SELECT fname, lname, clientid, value_occurrence
FROM
(SELECT  clientid, COUNT(clientid) AS value_occurrence
FROM  appointment
GROUP BY clientid
ORDER BY value_occurrence DESC) app
natural join client
natural join personX
```

	FNAME	LNAME	CLIENTID	VALUE_OCCURRENCE
1	Grace	Viterelli	1000	4
2	Sal	Wopat	1001	3
3	Fairuza	Fiennes	1002	3
4	Mike	Collins	1003	3
5	Milla	Phoenix	1007	3

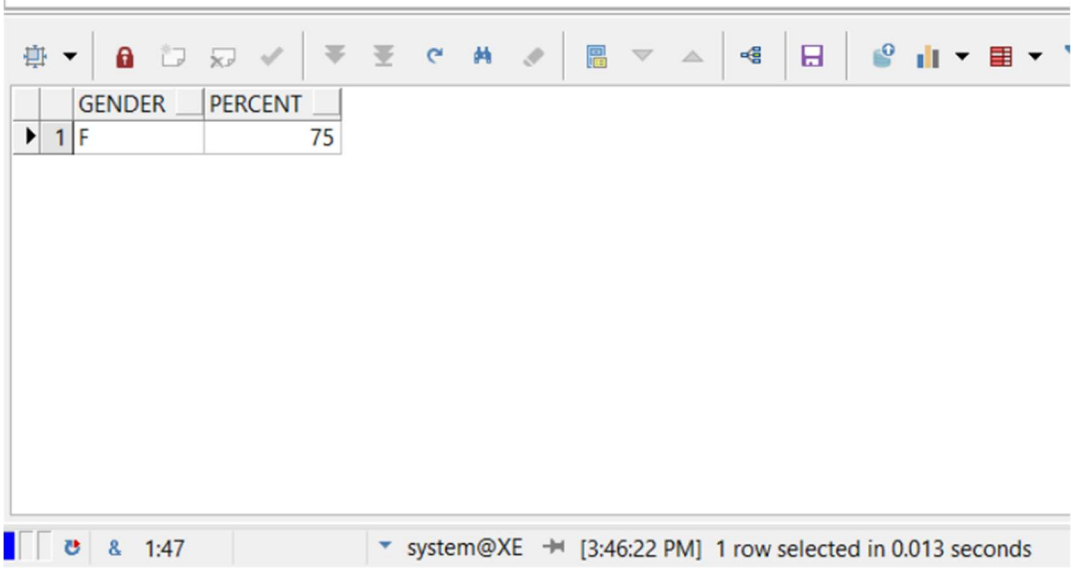
8:21 system@XE [3:25:46 PM] 5 rows selected in 0.019 seconds

Run time: 0.019 sec

2.

Calculates the percent of the women therapists (gender = 'F').

```
SELECT
gender, count(*)/(select count(*) from agent)*100 as percent
FROM
agent a natural join person p
group by gender
having gender = 'F'
```



Runtime: 0.13

3.

Calculating the total income of a given month from all treatments and agents

In the example we checked for the second month of 2020

```
select sum(price*income) as "Total amount:" from treatment
natural join
(select appointment.type, count(appointment.type) as income
from appointment left join treatment on appointment.type = treatment.type
where Extract(month from beginDateTime) = 2 and Extract(year from beginDateTime) = 2020
group by appointment.type)
```

	Total amount:
1	580

& 2:42 system@XE [6:31:30 PM] 1 row selected in 0.041 seconds

Run time :0.041

4.

Get the clients that owe money for appointments and the amounts

SQL Output Statistics

```
select clientid, sum(price) as NIS_OWED
from appointment natural join payment natural join client natural join treatment natural join person
where payed=0
group by clientid
ORDER BY NIS_OWED DESC
```

	CLIENTID	NIS_OWED
1	1001	200
2	1007	200
3	1003	160
4	1000	100
5	1002	100

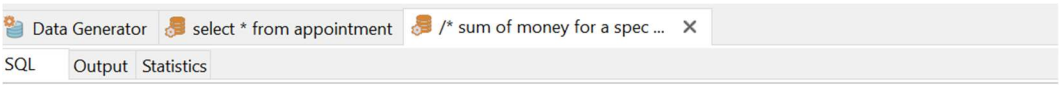
& 6:1 system@XE [3:40:15 PM] 5 rows selected in 0.027 seconds

Run time: 0.027

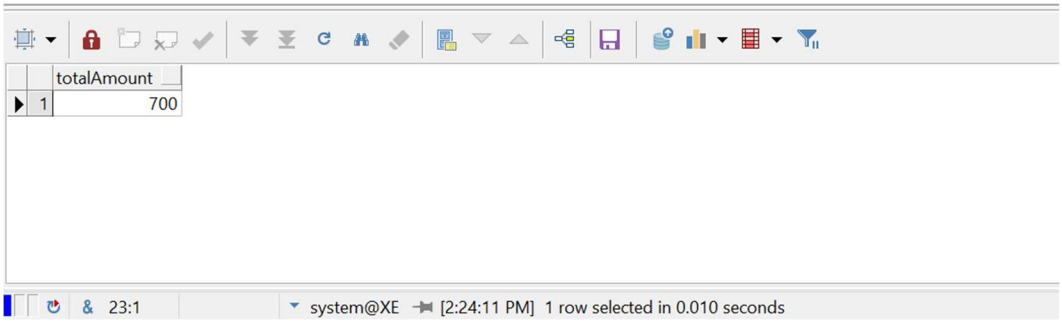
5.

Calculating the income from treatments for a specific therapist

In this example we checked for therapist who's id is 1000




```
select sum(treatment.price*NumberSessions) as "totalAmount"
from
treatment natural join (select type, count(begindatetime) as numberSessions
from appointment
where agentid=1000
group by type)
```

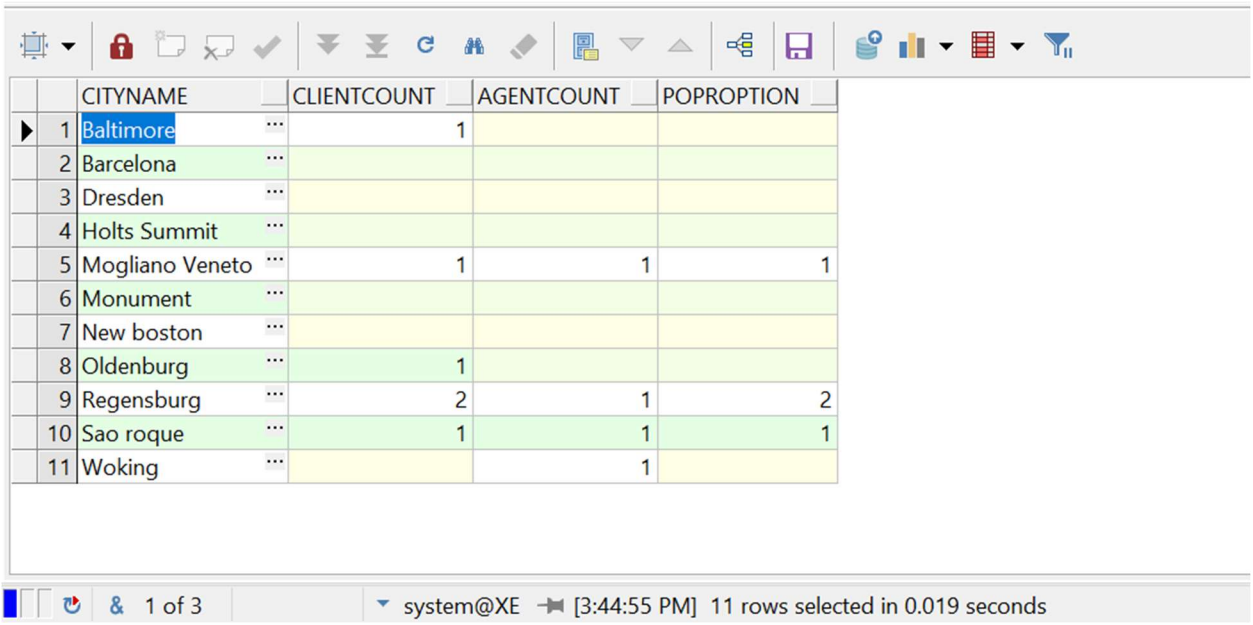


6.

Get the number of agents relational to number of clients for every city.

```
/*get the number of agents relational to number of clients in evrey city */
select cityname, clientCount, agentcount, clientCount/agentcount as poproption from city
left join
(select cityname as clientcity, count(distinct clientid) as clientCount
from client natural join person
group by cityname)
on clientcity = city.cityname
left join
(select cityname as agentcity, count(distinct agentid) as agentCount
from agent natural join person
group by cityname)
on agentcity = city.cityname
order by cityname
```

Result:



	CITYNAME	CLIENTCOUNT	AGENTCOUNT	POPROPTION
1	Baltimore	1		
2	Barcelona			
3	Dresden			
4	Holts Summit			
5	Mogliano Veneto	1	1	1
6	Monument			
7	New boston			
8	Oldenburg	1		
9	Regensburg	2	1	2
10	Sao roque	1	1	1
11	Woking		1	

Run time: 0.019 sec

7.

Indexes

We created these indexes:

We did on appointment table since it is one of the main tables we use, and a lot of the queries use this table, so we wanted to get the information quicker.

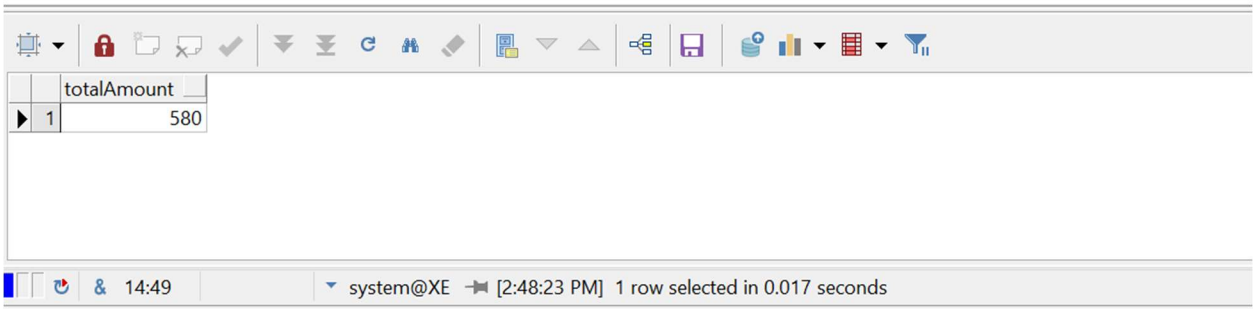
Index 1

CREATE INDEX appointment_begindatetime_i ON appointment(BEGINDATETIME);

We can see the improvement by executing query #3 after we have the index.

Runtime before: 0.041

```
select sum(treatment.price*income) as "totalAmount"
from treatment
natural join
(select appointment.type, count(appointment.type) as income
from appointment left join treatment on appointment.type = treatment.type
where Extract(month from beginDateTime) = 2 and Extract(year from beginDateTime) = 2020
group by appointment.type)
```

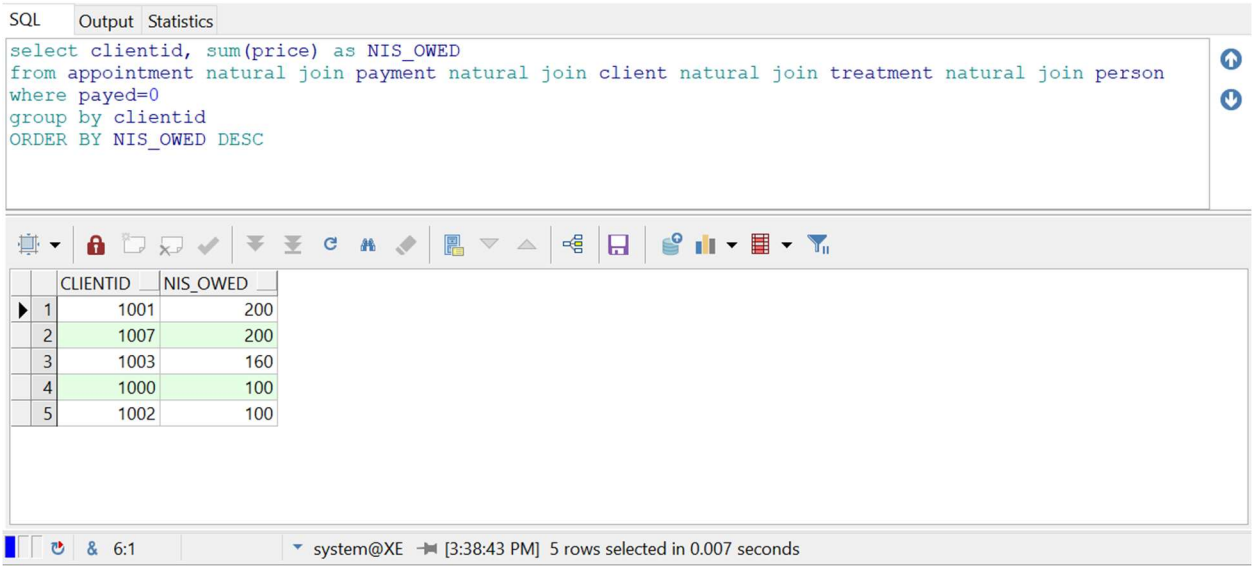


runtime with index: 0.017

Index 2

CREATE INDEX appointment_agentid_i ON appointment(agentid);

Runtime for query #4 after the index:



Runtime before the index: 0.0027

Runtime with index: 0.007

Index 3

CREATE INDEX appointment_clientid_i ON appointment(clientid);

Update queries:

1. To update the price for a specific treatment.

The treatment table:

	TYPE	PRICE	LENGTHTIME	DESCRIPTION
1	CBT	100	45	Congnitive Behavioral Therapy
2	DBT	200	80	Dialectic Behavioral Therapy
3	Psychodynamic	250	120	dyadic treatment
4	IFS	300	45	Internal Family Systems
5	HT	60	60	humanistic therapy
6	Animal	50	45	animal therapy
7	Art	100	60	art therapy
8	Music	200	60	music therapy

We want to update the price of CBT treatment to be 300.

```
update treatment
set price = 300
where TYPE = 'CBT';
```

	TYPE	PRICE	LENGTHTIME	DESCRIPTION
1	CBT	100	45	Congnitive Behavioral Therapy
2	DBT	200	80	Dialectic Behavioral Therapy
3	Psychodynamic	250	120	dyadic treatment
4	IFS	300	45	Internal Family Systems
5	HT	60	60	humanistic therapy
6	Animal	50	45	animal therapy
7	Art	100	60	art therapy
8	Music	200	60	music therapy

We wanted to update a person's address (people move sometimes 😉)

B4 the update

	FNAME	LNAME	BIRTHDATE	GENDER	ADDRESS	PHONENUM	EMAIL_	PERSONID	CITYNAME
1	Wang	Mifune	25-Dec-95	F	39 Or-yehuda Road	508447964	wang.mifune@eastmankodak.de	44573957	Holts Summit
2	Grace	Viterelli	19-Feb-77	M	36 Mazar Drive	550107496	grace.viterelli@vspan.uk	26338715	Baltimore
3	Sal	Wopat	24-Jan-88	M	76 Porter Blvd	505303612	salw@nha.com	55441950	Oldenburg
4	Fairuza	Fiennes	14-Mar-79	F	26 Austin Street	503826638	fairuza.fiennes@kingland.ca	32442690	Regensburg
5	Mike	Collins	19-Jun-85	M	1 Pottendorf Blvd	541057028	mike.c@taycorfinancial.br	79016729	Sao roque
6	Anna	Gaines	11-Feb-96	F	85 Ani Street	515451579	anna.gaines@bioreliance.si	27934537	Monument
7	Milla	Phoenix	08-Dec-72	F	27 Chloe Street	555975629	milla.p@bigdoughcom.at	13297832	Mogliano Veneto
8	Lorraine	Thompson	28-May-71	M	77 Haslam Street	513216219	lorraine.thompson@hewlettpacka	78614740	Barcelona
9	Kate	Conlee	11-Feb-02	M	16 Kenneth Street	522215558	kate.conlee@otbd.uk	64789966	Regensburg
10	Merrill	Henriksen	08-Aug-65	F	50 Lorenz Drive	532128083	merrill.henriksen@keith.com	91412598	Woking

After the update:

```
update person
set cityname = 'Barcelona',
ADDRESS =' 77 Haslam Street'
where personid= 44573957
```

	FNAME	LNAME	BIRTHDATE	GENDER	ADDRESS	PHONENUM	EMAIL_	PERSONID	CITYNAME
1	Wang	Mifune	25-Dec-95	F	77 Haslam Street	508447964	wang.mifune@eastmankodak.de	44573957	Barcelona
2	Grace	Viterelli	19-Feb-77	M	36 Mazar Drive	550107496	grace.viterelli@vspan.uk	26338715	Baltimore
3	Sal	Wopat	24-Jan-88	M	76 Porter Blvd	505303612	salw@nha.com	55441950	Oldenburg
4	Fairuza	Fiennes	14-Mar-79	F	26 Austin Street	503826638	fairuza.fiennes@kingland.ca	32442690	Regensburg
5	Mike	Collins	19-Jun-85	M	1 Pottendorf Blvd	541057028	mike.c@taycorfinancial.br	79016729	Sao roque
6	Anna	Gaines	11-Feb-96	F	85 Ani Street	515451579	anna.gaines@bioreliance.si	27934537	Monument
7	Milla	Phoenix	08-Dec-72	F	27 Chloe Street	555975629	milla.p@bigdoughcom.at	13297832	Mogliano Veneto
8	Lorraine	Thompson	28-May-71	M	77 Haslam Street	513216219	lorraine.thompson@hewlettpacka	78614740	Barcelona
9	Kate	Conlee	11-Feb-02	M	16 Kenneth Street	522215558	kate.conlee@otbd.uk	64789966	Regensburg
10	Merrill	Henriksen	08-Aug-65	F	50 Lorenz Drive	532128083	merrill.henriksen@keith.com	91412598	Woking

Delete

Before:

```
select * from treatment
```

	TYPE	PRICE	LENGTHTIME	DESCRIPTION
1	CBT	100	45	Congnitive Behavioral Therapy
2	DBT	200	80	Dialectic Behavioral Therapy
3	Psychodynamic	250	120	dyadic treatment
4	IFS	300	45	Internal Family Systems
5	HT	60	60	humanistic therapy
6	Animal	50	45	animal therapy
7	Art	100	60	art therapy
8	Music	200	60	music therapy

We wanted to delete a treatment from our services such as ‘music’

```
--select * from treatment
delete from treatment where type = 'Music '
```

The results:

```
select * from treatment
--delete from treatment where type = 'Music '
```

	TYPE	PRICE	LENGTHTIME	DESCRIPTION
1	CBT	100	45	Congnitive Behavioral Therapy
2	DBT	200	80	Dialectic Behavioral Therapy
3	Psychodynamic	250	120	dyadic treatment
4	IFS	300	45	Internal Family Systems
5	HT	60	60	humanistic therapy
6	Animal	50	45	animal therapy
7	Art	100	60	art therapy

To delete a city – we stopped working there cuz people there are too mentally healthy …

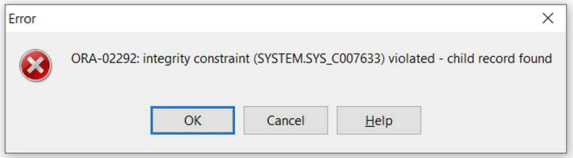
```
delete from city where cityname = 'Bad Camberg'
```

```
--delete from city where cityname = 'Bad Camberg'
select * from city
```

	CITYNAME	AREAID
1	Monument	1
2	Mogliano Veneto	2
3	New boston	2
4	Barcelona	2
5	Holts Summit	2
6	Woking	3
7	Regensburg	3
8	Oldenburg	4
9	Baltimore	4
10	Sao roque	4
11	Dresden	4

Attempting to delete a field that is a foreign key in other tables. In the example we tried to delete the treatment ‘HT’ but since there ‘HT’ appointments so it’s not possible.

```
--select * from treatment
delete from treatment where type ='HT'
```



Commit & rollback:

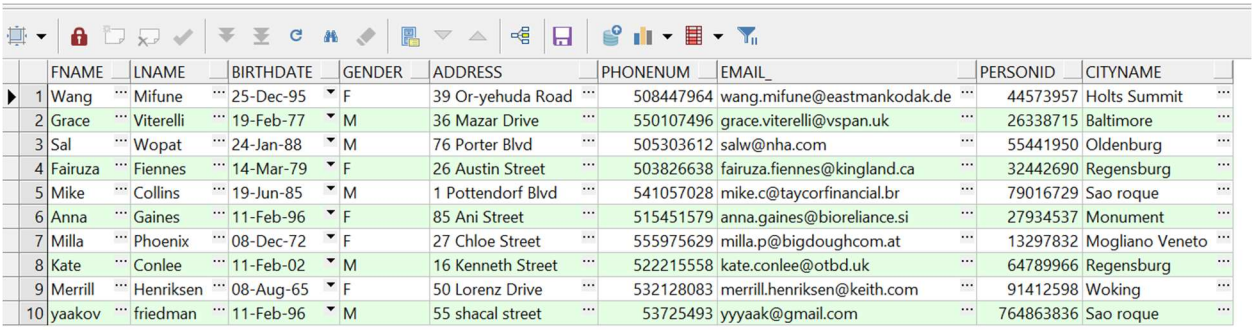
In the end of every session the system automatically commits the changes unless there was some error.

We'll show an example of a rollback:

We wanted to add a person to the system. In sql:

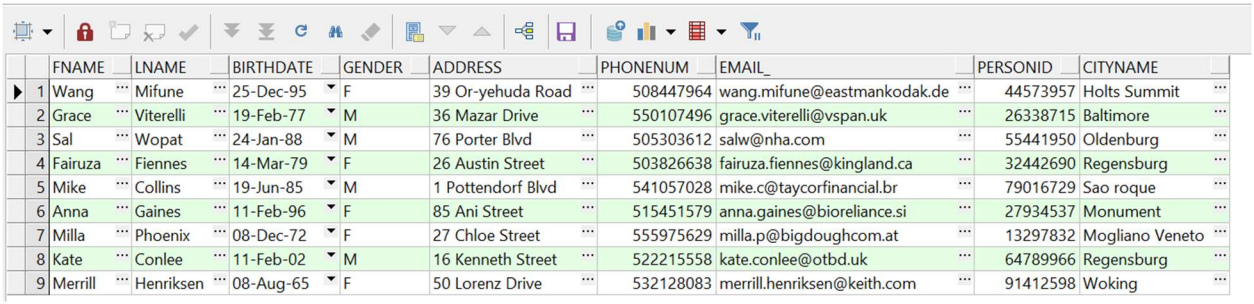
```
insert into person
values ('yaakov', 'friedman', '11-Feb-96','M', '55 shacal
street','053725493', 'yyyaak@gmail.com', 764863836, 'Sao roque')
```

this is what it did:



	FNAME	LNAME	BIRTHDATE	GENDER	ADDRESS	PHONENUM	EMAIL_	PERSONID	CITYNAME
1	Wang	Mifune	25-Dec-95	F	39 Or-yehuda Road	508447964	wang.mifune@eastmankodak.de	44573957	Holts Summit
2	Grace	Viterelli	19-Feb-77	M	36 Mazar Drive	550107496	grace.viterelli@vspan.uk	26338715	Baltimore
3	Sal	Wopat	24-Jan-88	M	76 Porter Blvd	505303612	salw@nha.com	55441950	Oldenburg
4	Fairuza	Fiennes	14-Mar-79	F	26 Austin Street	503826638	fairuza.fiennes@kingland.ca	32442690	Regensburg
5	Mike	Collins	19-Jun-85	M	1 Pottendorf Blvd	541057028	mike.c@taycorfinancial.br	79016729	Sao roque
6	Anna	Gaines	11-Feb-96	F	85 Ani Street	515451579	anna.gaines@bioreliance.si	27934537	Monument
7	Milla	Phoenix	08-Dec-72	F	27 Chloe Street	555975629	milla.p@bigdoughcom.at	13297832	Mogliano Veneto
8	Kate	Conlee	11-Feb-02	M	16 Kenneth Street	522215558	kate.conlee@otbd.uk	64789966	Regensburg
9	Merrill	Henriksen	08-Aug-65	F	50 Lorenz Drive	532128083	merrill.henriksen@keith.com	91412598	Woking
10	yaakov	friedman	11-Feb-96	M	55 shacal street	53725493	yyyaak@gmail.com	764863836	Sao roque

After rollback:



	FNAME	LNAME	BIRTHDATE	GENDER	ADDRESS	PHONENUM	EMAIL_	PERSONID	CITYNAME
1	Wang	Mifune	25-Dec-95	F	39 Or-yehuda Road	508447964	wang.mifune@eastmankodak.de	44573957	Holts Summit
2	Grace	Viterelli	19-Feb-77	M	36 Mazar Drive	550107496	grace.viterelli@vspan.uk	26338715	Baltimore
3	Sal	Wopat	24-Jan-88	M	76 Porter Blvd	505303612	salw@nha.com	55441950	Oldenburg
4	Fairuza	Fiennes	14-Mar-79	F	26 Austin Street	503826638	fairuza.fiennes@kingland.ca	32442690	Regensburg
5	Mike	Collins	19-Jun-85	M	1 Pottendorf Blvd	541057028	mike.c@taycorfinancial.br	79016729	Sao roque
6	Anna	Gaines	11-Feb-96	F	85 Ani Street	515451579	anna.gaines@bioreliance.si	27934537	Monument
7	Milla	Phoenix	08-Dec-72	F	27 Chloe Street	555975629	milla.p@bigdoughcom.at	13297832	Mogliano Veneto
8	Kate	Conlee	11-Feb-02	M	16 Kenneth Street	522215558	kate.conlee@otbd.uk	64789966	Regensburg
9	Merrill	Henriksen	08-Aug-65	F	50 Lorenz Drive	532128083	merrill.henriksen@keith.com	91412598	Woking

We can see that it didn't take into account the last sql statement (adding Yaakov into the system).

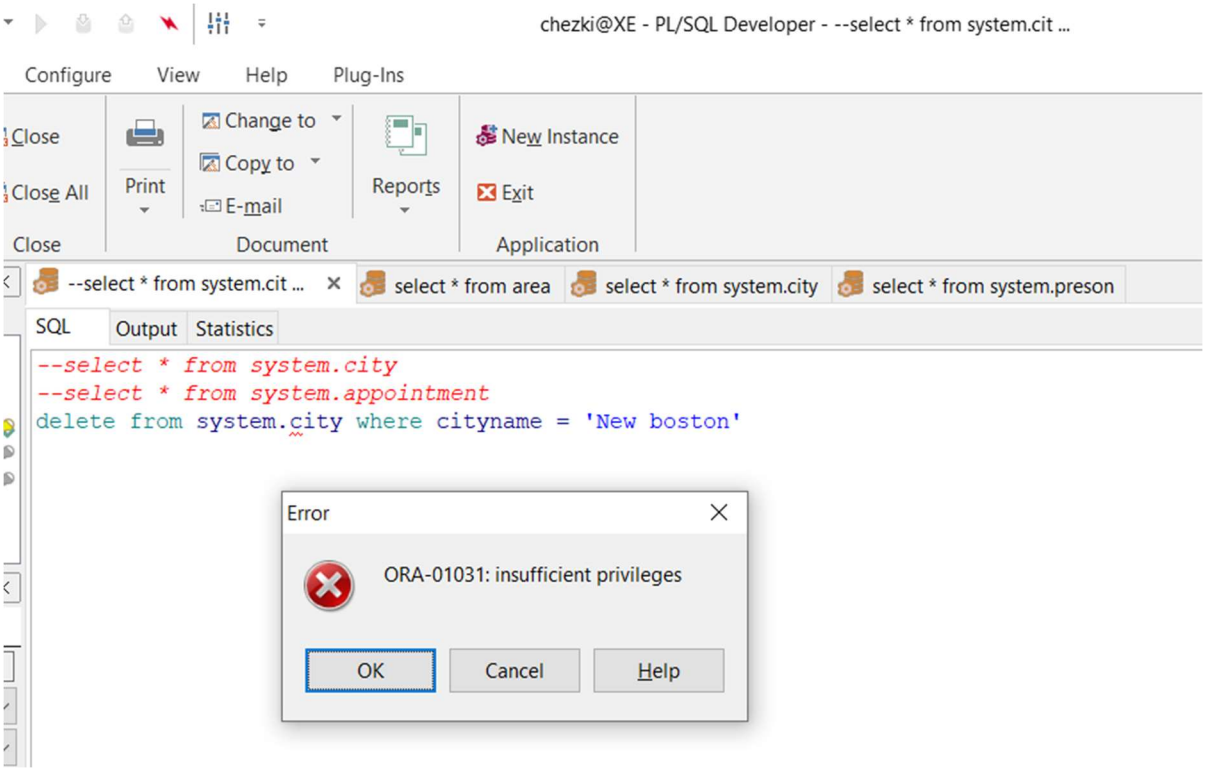
Users

We added a new user to the system as follows:

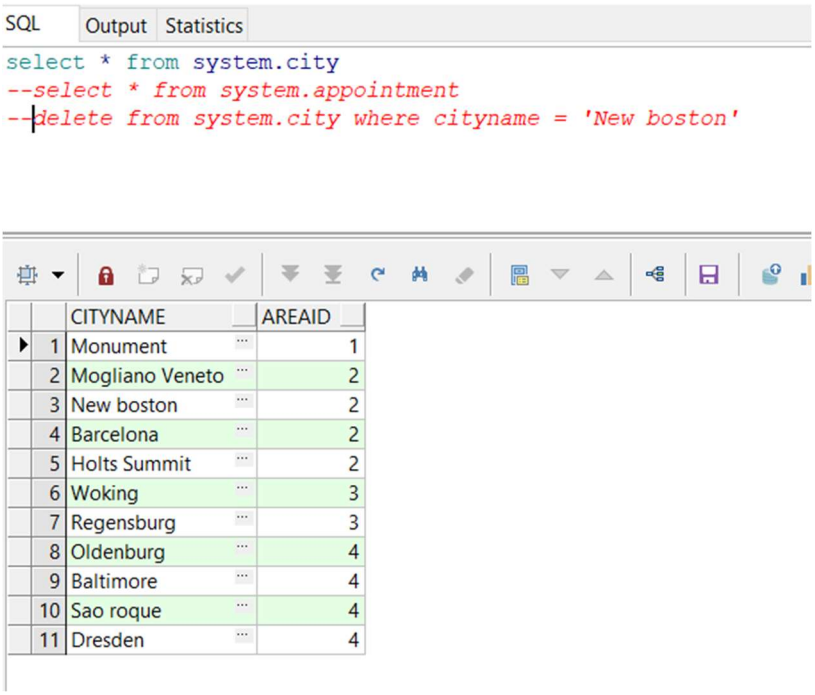
```
alter session set "_ORACLE_SCRIPT"=true;
create user chezki --user name
IDENTIFIED BY 1234 --passward
default tablespace SYSTEM
temporary tablespace TEMP
profile DEFAULT
password expire;

grant update on city to chezki ;
grant select on city to chezki ;
grant create session to chezki ; --להתחבר הרשאה
--grant create table to chezki ; -- טבלאות ליצירת הרשאה
```

When the user is logged-in we can see that It can't access/change things he doesn't have permission (we can see on top that chezki is logged in – "chezki@xe").

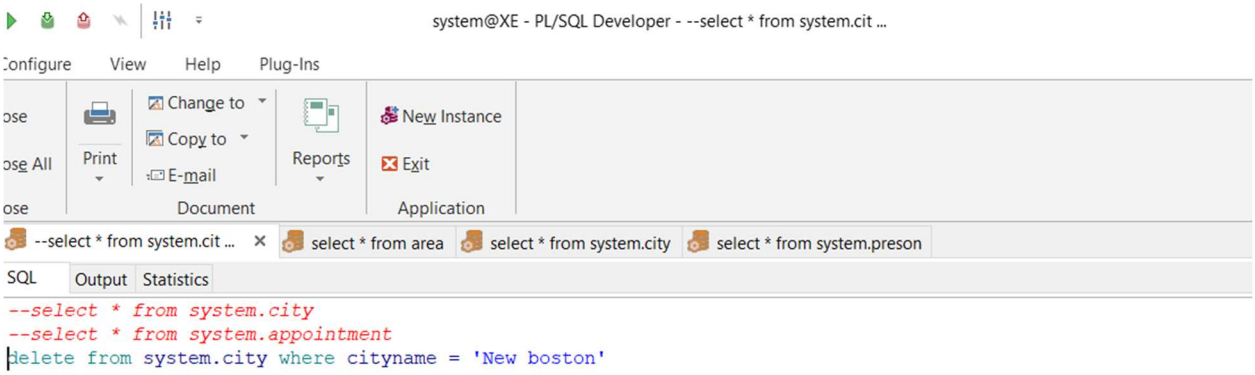


However chezki has access to 'select' commands:



	CITYNAME	AREAID
1	Monument	1
2	Mogliano Veneto	2
3	New boston	2
4	Barcelona	2
5	Holts Summit	2
6	Woking	3
7	Regensburg	3
8	Oldenburg	4
9	Baltimore	4
10	Sao roque	4
11	Dresden	4

however a user that has permission can make the changes. For example 'system' tries to delete – the same script 'chezki' was rejected:



Constraints

```
ALTER TABLE treatment
MODIFY lengthtime DEFAULT 45;
```

Default for length time to be 45.

If we try to insert a new treatment "demo" without default like this:

```
insert into treatment (TYPE, PRICE, DESCRIPTION)
VALUES ('MM', 300, 'demo treatment');
```

We can see it inserted the 45.

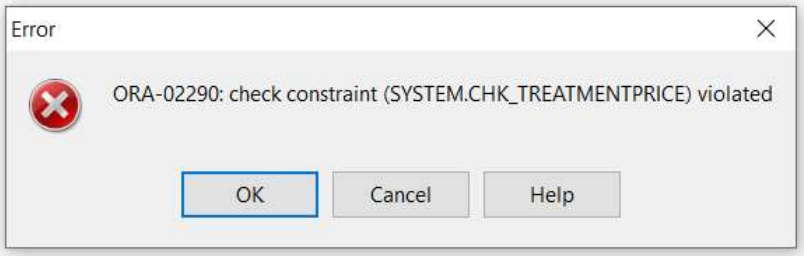
	TYPE	PRICE	LENGTHTIME	DESCRIPTION
1	MM	300	45	demo treatment
2	CBT	100	45	Congnitive Behavioral Therapy
3	DBT	200	80	Dialectic Behavioral Therapy
4	Psychodynamic	250	120	dyadic treatment
5	IFS	300	45	Internal Family Systems
6	HT	60	60	humanistic therapy
7	Animal	50	45	animal therapy
8	Art	100	60	art therapy

A constraint for a range in price for a treatment:

```
ALTER TABLE Treatment
ADD CONSTRAINT CHK_TreatmentPrice CHECK (price>=30 AND price<=500);
```

If we insert with a value that isn't in the range:

```
insert into treatment (type, price, lengthtime, description)
values ('GG', 1000, 100, 'demo');
```



We can see there is an error since it violates the constraints.

Views

A view that shows the occupied appointments.

```
CREATE VIEW BusyTimes AS
select beginDateTime, LengthTime
from appointment natural join Treatment
```

Result:

```
select * from busyTimes
```

	BEGINDATETIME	LENGTHTIME
1	1/10/2020 11:30:00 AM	45
2	1/13/2020 3:00:00 PM	45
3	2/15/2020 6:15:00 PM	80
4	2/28/2020 6:45:00 AM	60
5	1/15/2020 1:30:00 PM	45
6	1/3/2020 8:15:00 PM	60
7	2/10/2020 4:30:00 AM	60
8	1/4/2020 5:15:00 PM	60
9	1/30/2020 6:30:00 PM	45
10	1/15/2020 1:15:00 PM	80
11	1/19/2020 5:45:00 PM	60
12	2/13/2020 11:15:00 PM	45
13	2/8/2020 11:30:00 PM	60
14	2/9/2020 2:00:00 PM	45
15	1/5/2020 4:00:00 PM	80
16	1/8/2020 2:15:00 PM	45

Another view for the clients in a specific city. Can be very useful with sending messages for important updates.

In the example for “Refensburg” clients.

```
CREATE VIEW RegensburgClients AS
select fName,lname, email_
from client natural join person
where cityName = 'Regensburg'
```

Result:

```
select * from RegensburgClients
```

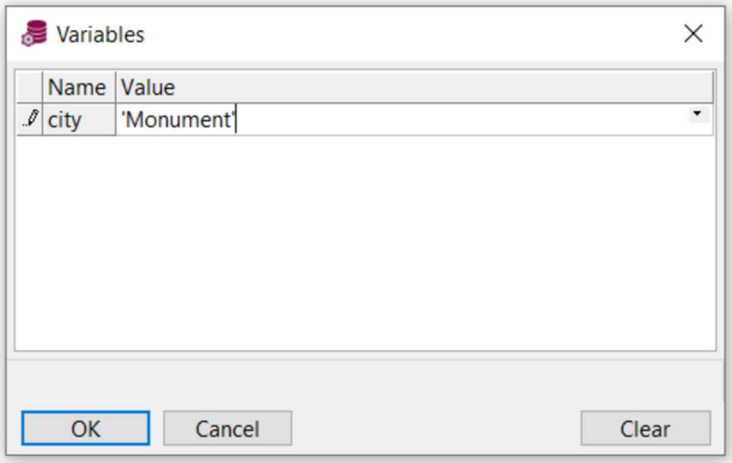
	FNAME	LNAME	EMAIL_
1	Fairuza	Fiennes	fairuza.fiennes@kingland.ca
2	Kate	Conlee	kate.conlee@otbd.uk

Part 3

Queries with parameters:

A.

```
select * from person where cityname = &city
```



Results in:

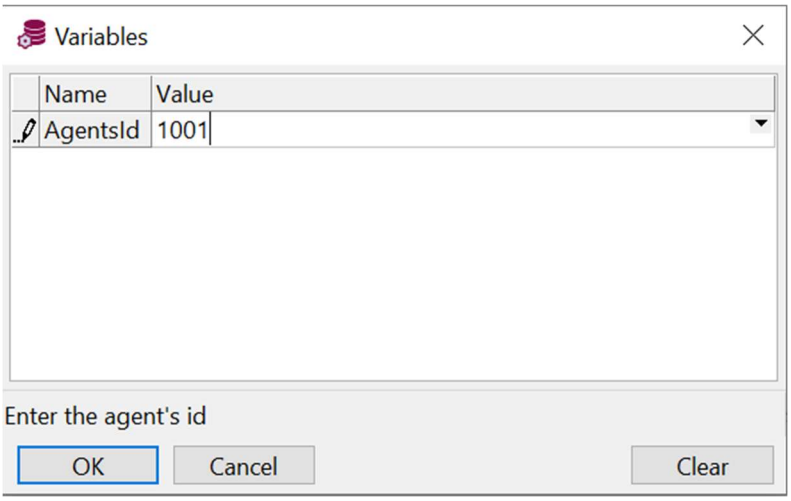
```
select * from person where cityname = &city
```

	FNAME	LNAME	BIRTHDATE	GENDER	ADDRESS	PHONENUM	EMAIL_	PERSONID	CITYNAME
1	Wang	Mifune	12/25/1995	F	39 Or-yehuda Road	508447964	wang.mifune@eastmankodak.de	44573957	Monument
2	Anna	Gaines	2/11/1996	F	85 Ani Street	515451579	anna.gaines@bioreliance.si	27934537	Monument

B.

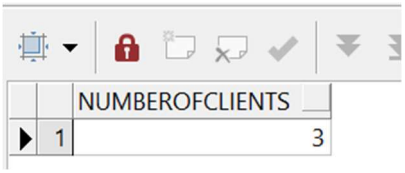
The following query prompts to input an agent id and returns the number of clients the agent has.

```
select numberOfClients from
(
select agentid, count(clientid) as numberOfClients
from
appointment group by agentid
)
where agentid = &<name = "AgentsId" hint = "Enter the agent's id" type = "integer">
```



If we enter for example 1001 as input.

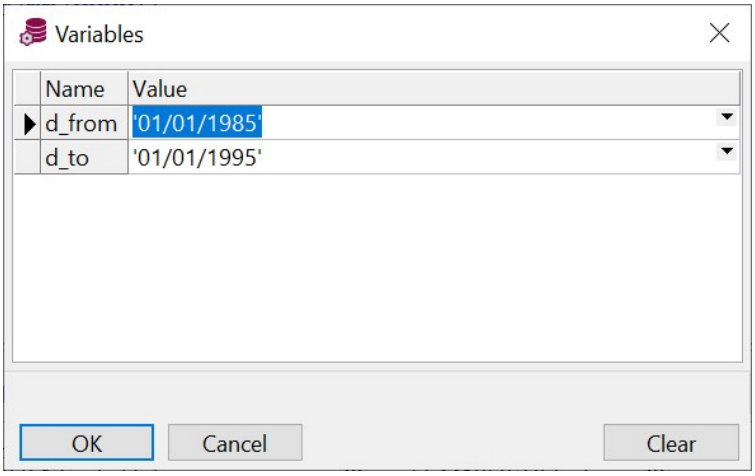
The result:



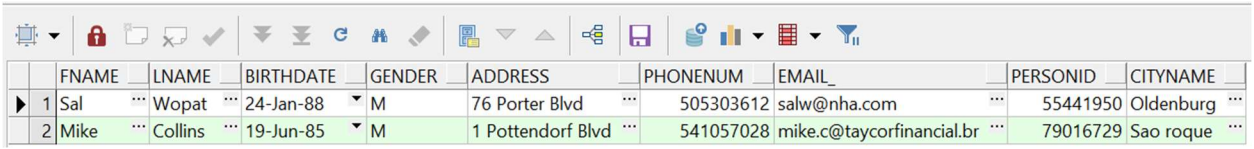
C.

Query to get the people in the system that were born between specific date.

```
select * from person
where birthdate
between to_date(&d_from, 'dd/mm/yyyy') and to_date(&d_to,'dd/mm/yyyy')
```



Result:



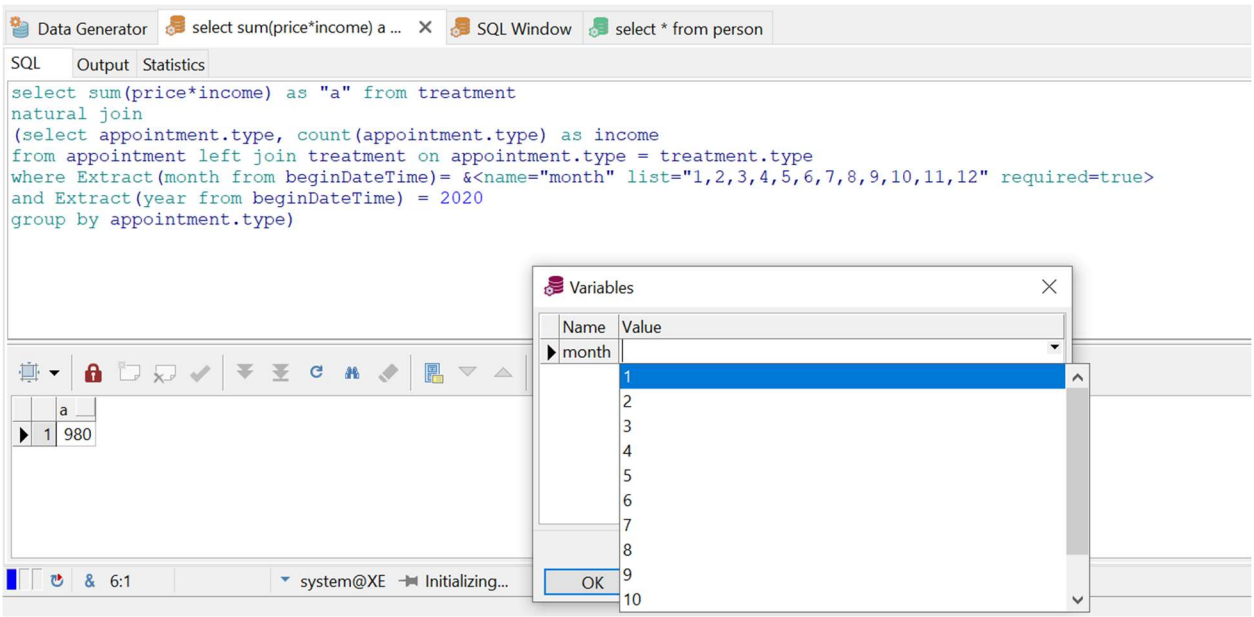
	FNAME	LNAME	BIRTHDATE	GENDER	ADDRESS	PHONENUM	EMAIL_	PERSONID	CITYNAME
1	Sal	Wopat	24-Jan-88	M	76 Porter Blvd	505303612	salw@nha.com	55441950	Oldenburg
2	Mike	Collins	19-Jun-85	M	1 Pottendorf Blvd	541057028	mike.c@taycorfinancial.br	79016729	Sao roque

These are the people that were born between these dates.

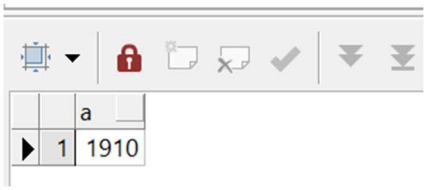
D.

The next query calculates the total income for a given month.

We choose from a list of the months in the year.



If we choose January (1) then the result is:



	a
1	1910

Reports

A.

Report for all the booked appointments:

```
select begindatetime, lengthtime
from appointment
natural join treatment
```

Begindatetime	Lengthtime
1/10/2020 11:30:00 AM	45
1/13/2020 3:00:00 PM	45
2/15/2020 6:15:00 PM	80
2/28/2020 6:45:00 AM	60
1/15/2020 1:30:00 PM	45
1/3/2020 8:15:00 PM	60
2/10/2020 4:30:00 AM	60
1/4/2020 5:15:00 PM	60
1/30/2020 6:30:00 PM	45
1/15/2020 1:15:00 PM	80
1/19/2020 5:45:00 PM	60
2/13/2020 11:15:00 PM	45
2/8/2020 11:30:00 PM	60
2/9/2020 2:00:00 PM	45
1/5/2020 4:00:00 PM	80
1/8/2020 2:15:00 PM	45

We can see the begin of the appointments and the length of it.

It is very useful.

B.

Report that shows the people in the system that live in a specific city.

The city is given as a parameter from the user.

```
select fName,lname, email_
from client natural join person
where cityName = &cityName
```

cityName = 'Regensburg'

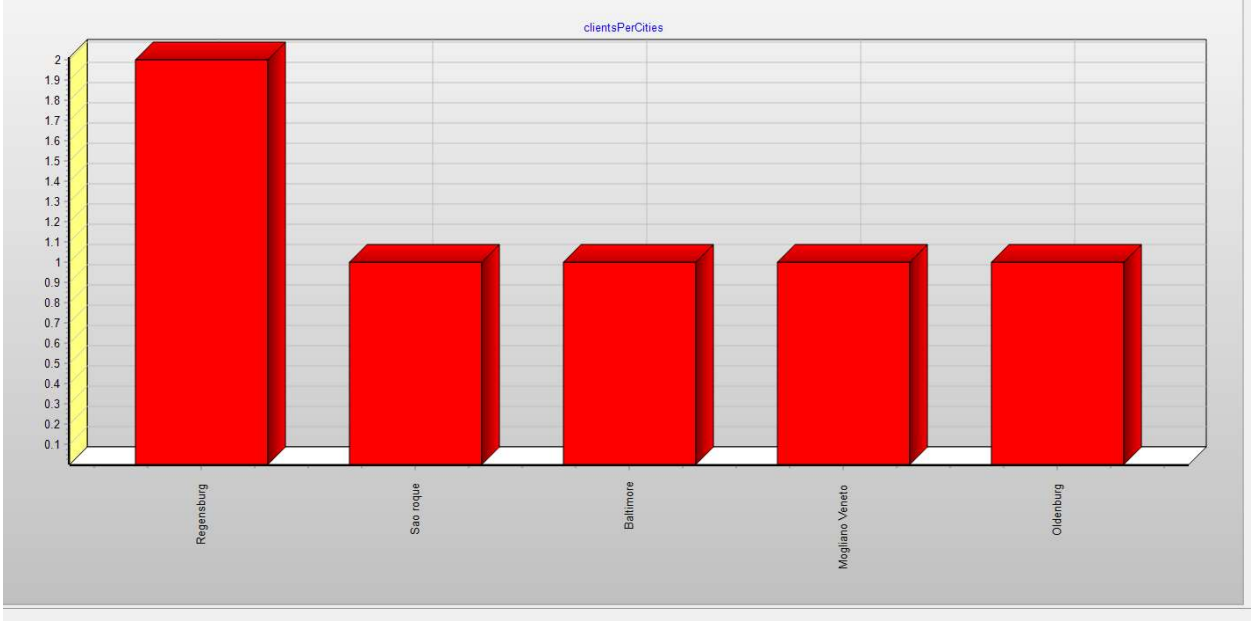
Fname	Lname	Email_
Fairuza	Fiennes	fairuza.fiennes@kingland.ca
Kate	Conlee	kate.conlee@otbd.uk

Graphs

```
select cityName, count(fName)
from client natural join person
group by person.cityName;
```

Cityname	Count(fname)
Regensburg	2
Sao roque	1
Baltimore	1
Mogliano Veneto	1
Oldenburg	1

For this report that gets the number of clients per city, we created a graph:



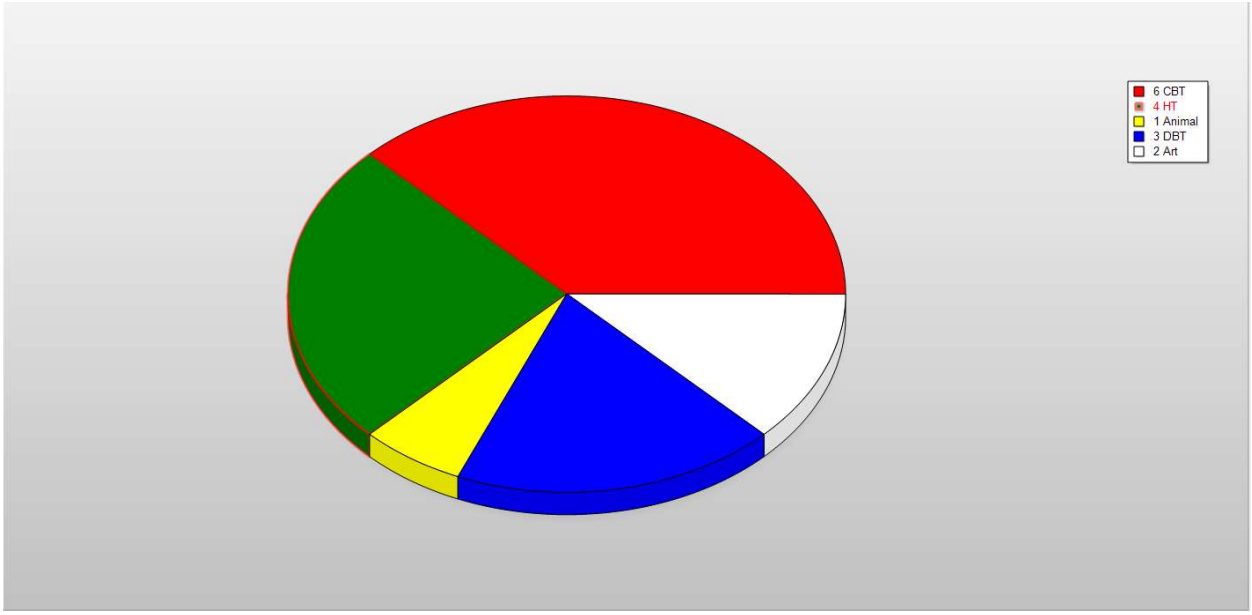
X axis is for the name of the cities.

Y axis is for the number of clients in each city.

2.

The following graph shows the booked appointments according to the type of treatment.

It is useful to know what's the most popular type of appointments.



Functions

1.

Function that calculates the number of clients per given agent. Assume a specific client has 3 appointments he will still be counted once.

Input: agent id.

Output: number of clients the agent has.

```
create or replace function count_client_per_agent(idagent in number) return
integer is
  result integer;
begin
     select count(clientid) into result
  from
     (select DISTINCT clientid
      from appointment
      where appointment.agentid=idagent);
      return (result);

end count_client_per_agent;
```

example:

```
DECLARE
a number;


begin
a:= count_client_per_agent(1001);
dbms_output.put_line(a);
end;
```

Result:

```
4 clients
```

2.

We wanted to make sure that when adding a new appointment, the client and the therapist are available at that time and neither of them are busy with a different session. In order to achieve this we first wrote a function that checks for given date, clientid, agentid if they are available at that time to set that appointment.

Only if they are available we can proceed and add the appointment.

This function does that:

```
create or replace function check_if_available
(input_agent in number, input_client in number, input_date in date)
return boolean
is
c_id  number;
a_id  number;
b_date date;
h_date VARCHAR(20);
d_date VARCHAR(20);
in_hour varchar(20);
in_date varchar(20);
cursor booked_appointmens is select clientid, agentid, begindatetime
from
appointment;
begin
in_hour:=to_char(input_date,'HH24:MI');
in_date:=to_char(input_date,'DD-Mon-YYYY' );
dbms_output.put_line('input date and time: '||in_date||' ' ||in_hour);
open booked_appointmens;
loop
  fetch booked_appointmens into c_id, a_id, b_date;
        EXIT WHEN booked_appointmens%notfound;
h_date:=to_char(b_date,'HH24:MI');
d_date:=to_char(b_date,'DD-Mon-YYYY' );
if in_hour=h_date and in_date=d_date
then
  if c_id=input_client
    then
      dbms_output.put_line('client has another session!');
      return false;
    else
      if a_id=input_agent
        then
         dbms_output.put_line('therapist has another session!');
         return false;
         end if;
  end if;
end if;
end loop;
close booked_appointmens;
return true;
end check_if_available;
```

3.

Calculates income for a specific agent;

Input: agent id.

Output: total income from all sessions.

```
CREATE OR REPLACE function Calc_income_agent(agent_id in number) return
integer is
 result integer;

begin

select sum(treatment.price*NumberSessions) into result
from
treatment natural join( select type, count(begindatetime) as numberSessions
from appointment
where agentid=agent_id
group by type);

return (result);

end Calc_income_agent;
```

Example:

```
declare
a integer;
begin

a:=calc_income_agent(1001);
dbms_output.put_line('Total income: '||a);


end;
```

Result:

```
Total income: 310
```

Procedures

1.

Increases percentage of price for a treatment.

Input: the treatment, and percentage.

Updates the price.

```
CREATE OR REPLACE PROCEDURE update_percent(treat_type IN varchar, percentage
in number) is
begin
update treatment
set price = price*(percentage+100)/100
where type=treat_type;
end;
```

example:

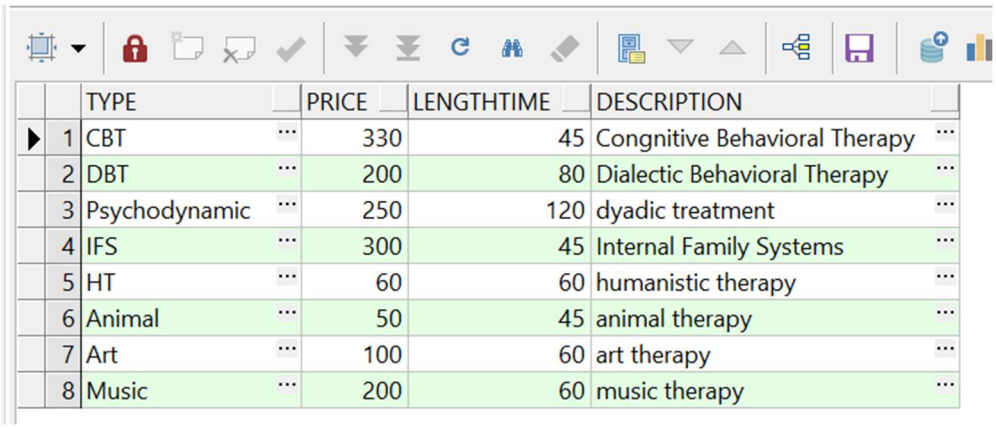
```
begin

update_percent('CBT', 10);

end;
```

```
(the price for CBT treatment was 300)
```

```
this will result in:
```



	TYPE	PRICE	LENGTHTIME	DESCRIPTION
1	CBT	330	45	Congnitive Behavioral Therapy
2	DBT	200	80	Dialectic Behavioral Therapy
3	Psychodynamic	250	120	dyadic treatment
4	IFS	300	45	Internal Family Systems
5	HT	60	60	humanistic therapy
6	Animal	50	45	animal therapy
7	Art	100	60	art therapy
8	Music	200	60	music therapy

2.

shows the income for every agent:

input: agent id.

Output: prints all agents and the corresponding income for each one.

```
CREATE OR REPLACE PROCEDURE show_income_of_all_agents
is
   a_proffesion agent.proffesion%type;
   city_name person.cityname%type;
   a_id person.personid%type;
   f_name person.fname%type;
   l_name person.lname%type;
   income number;
   CURSOR a_agents is
      SELECT agentid, fname, lname, proffesion, cityname FROM agent natural join person;
BEGIN
   OPEN a_agents;
   LOOP
   FETCH a_agents into a_id, f_name, l_name, a_proffesion, city_name;
      EXIT WHEN a_agents%notfound;
      income:=Calc_income_agent(a_id);
      dbms_output.put_line(a_id || ', ' || f_name || ', ' || l_name||', '|| a_proffesion|| ', '
      ||city_name||' ---> income: '||income);
   END LOOP;
   CLOSE a_agents;
END show_income_of_all_agents;
```

When running it:

```
begin
 show_income_of_all_agents;

 end;
```

Result:

```
1001, Fairuza, Fiennes, therapist, Regensburg ---> income: 310
1002, Mike, Collins, psycholog, Sao roque ---> income: 450
1000, Milla, Phoenix, therapist, Mogliano Veneto ---> income: 1620
1003, Merrill, Henriksen, psycholog, Woking ---> income: 690
```

3.

Shows all the sessions that a client didn't pay:

Input: id of a client

Output: prints all the sessions that didn't pay with the amount.

```
CREATE OR REPLACE PROCEDURE sessions_not_payed(c_id in number)
is
   client_id person.personid%type;
   f_name person.fname%type;
   l_name person.lname%type;
   a_date appointment.begindatetime%type;
   agent_id appointment.agentid%type;
   did_pay payment.agentid%type;
   t_price treatment.price%type;

   CURSOR client_appointments is
      SELECT personid,  fname, lname, begindatetime, agentid, payed, price
        FROM client natural join appointment natural join person natural join
payment natural join treatment;
begin
   OPEN client_appointments;
   LOOP
   FETCH client_appointments into client_id, f_name, l_name, a_date,
agent_id, did_pay, t_price;
      EXIT WHEN client_appointments%notfound;

      if client_id=c_id and did_pay = 0
        then
     dbms_output.put_line(f_name ||' ' || l_name ||'(id:'||client_id||')' ||'
--> On the: ' || a_date ||', Had session with Therapist id: ' ||agent_id||'
and owes: '||t_price);
     end if;

 end loop;

 CLOSE client_appointments;

 end sessions_not_payed;
```

running example:

```
begin

sessions_not_payed(26338715);

end;
```

Result:

```
Grace Viterelli(id:26338715) --> On the: 08-JAN-20, Had session with Therapist id: 1003 and owes: 330
```

Triggers

```
CREATE OR REPLACE TRIGGER ON_APPOINTMENT_CHANGE
       AFTER INSERT OR UPDATE OR DELETE
       ON APPOINTMENT
BEGIN
  DBMS_OUTPUT.PUT_LINE('appointment changed!');
END;
```

This is a trigger for changing any information of an appointment to print a message that the appointment changed.

Example:

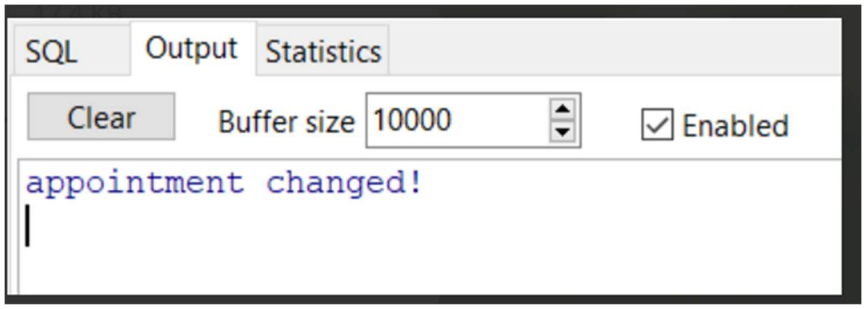
```
update appointment
set agentid = 1002
where agentid = 1000 and clientid = 1003 and type = 'art'
```

Now the table would look like this:

```
select * from appointment
```

	BEGINDATETIME	AGENTID	TYPE	CLIENTID
1	1/10/2020 11:30:00 AM	1000	CBT	1002
2	1/13/2020 3:00:00 PM	1001	Animal	1000
3	2/15/2020 6:15:00 PM	1003	DBT	1000
4	2/28/2020 6:45:00 AM	1001	HT	1002
5	1/15/2020 1:30:00 PM	1000	CBT	1007
6	1/3/2020 8:15:00 PM	1000	Art	1003
7	2/10/2020 4:30:00 AM	1002	HT	1003
8	1/4/2020 5:15:00 PM	1002	HT	1000
9	1/30/2020 6:30:00 PM	1002	CBT	1002
10	1/15/2020 1:15:00 PM	1001	DBT	1007
11	1/19/2020 5:45:00 PM	1003	Art	1007
12	2/13/2020 11:15:00 PM	1000	CBT	1001
13	2/8/2020 11:30:00 PM	1003	HT	1003
14	2/9/2020 2:00:00 PM	1000	CBT	1001
15	1/5/2020 4:00:00 PM	1000	DBT	1001
16	1/8/2020 2:15:00 PM	1003	CBT	1000

We can see the result of the trigger:



2.

Using function 2 (from before) we can add a trigger for when inserting into appointment to check if it is ok. If the previous function returns 'true' – it means that neither the client or the therapist have other session and we can book an appointment.

Otherwise, we cannot proceed.

The trigger:

```
create or replace trigger check_valid_appintment
before insert on appointment
for each row
declare
exp exception;
begin
if not (check_if_available(:new.agentid,:new.clientid, :new.begindatetime))
  then raise exp;
  end if;
exception
    when exp
      then raise_application_error(-20000,'cannot add to appointment - client
or therpaist have another session');
end check_valid_appintment;
```

example:

looking at the appointment table:

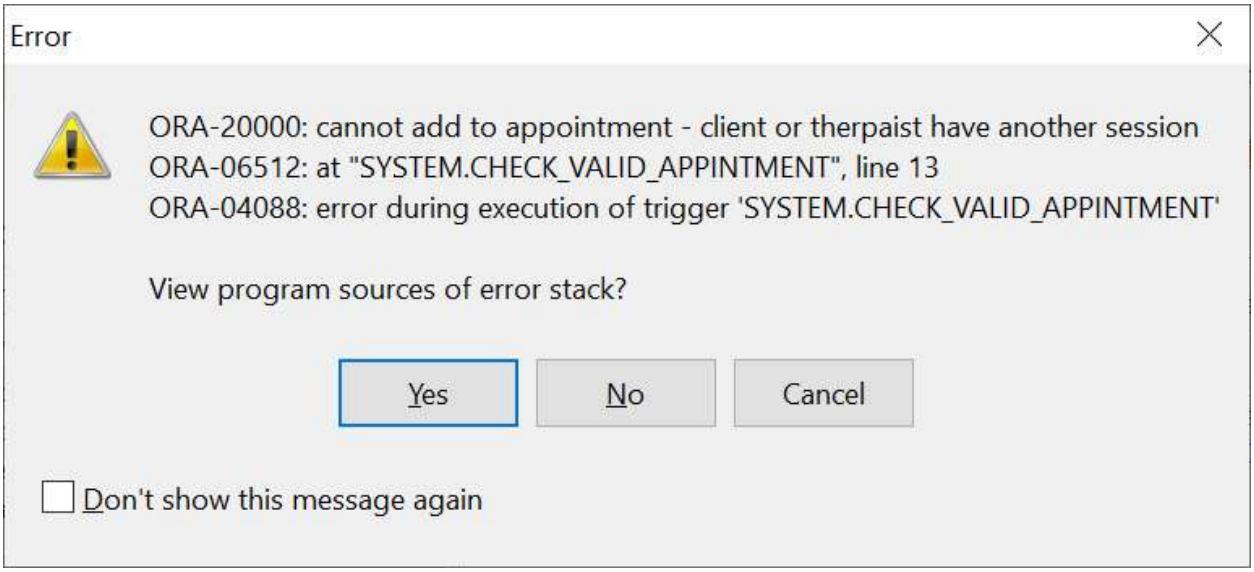
	BEGINDATETIME	AGENTID	TYPE	CLIENTID
1	10-Jan-20 11:30:00 AM	1000	CBT	1002
2	13-Jan-20 3:00:00 PM	1001	Animal	1000

Let's try and add the an appointment for the same time as the first with the same agentid and a different client (this isn't supposed to happen for the same agent to have a session with two different clients).

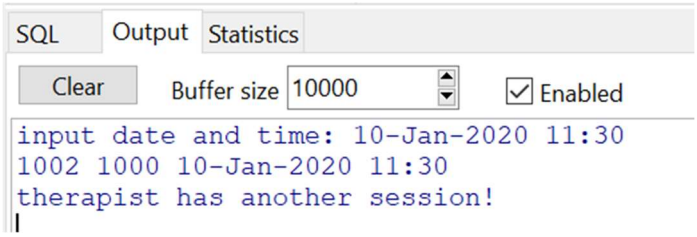
SQL Output Statistics

```
insert into appointment(begindatetime,agentid,type, clientid)
values(to_date('10-JAN-2020 11:30:00','DD-Mon-YYYY HH24:MI:ss'),1000,'CBT',1004);
```

This is what happens:



We can also see the output:



Where we can see that the problem is that the “therapist has another session!”

Part 4

We decided to add an option for sending newsletters to clients. Especially these days with social distancing it is very important to stay connected.

We added entity 'newsletter' that has the following attributes:

Newsletterid- id of the newsletter.

firstTime – the first time the newsletter was sent. It is also good for seeing when it started and relevant for next field that is:

frequency – for the frequency the newsletter is sent.

Topic – topic of the newsletter.

Agentid – the therapist that writes the newsletter.

In order to subscribe to newsletters, there is another entity named:

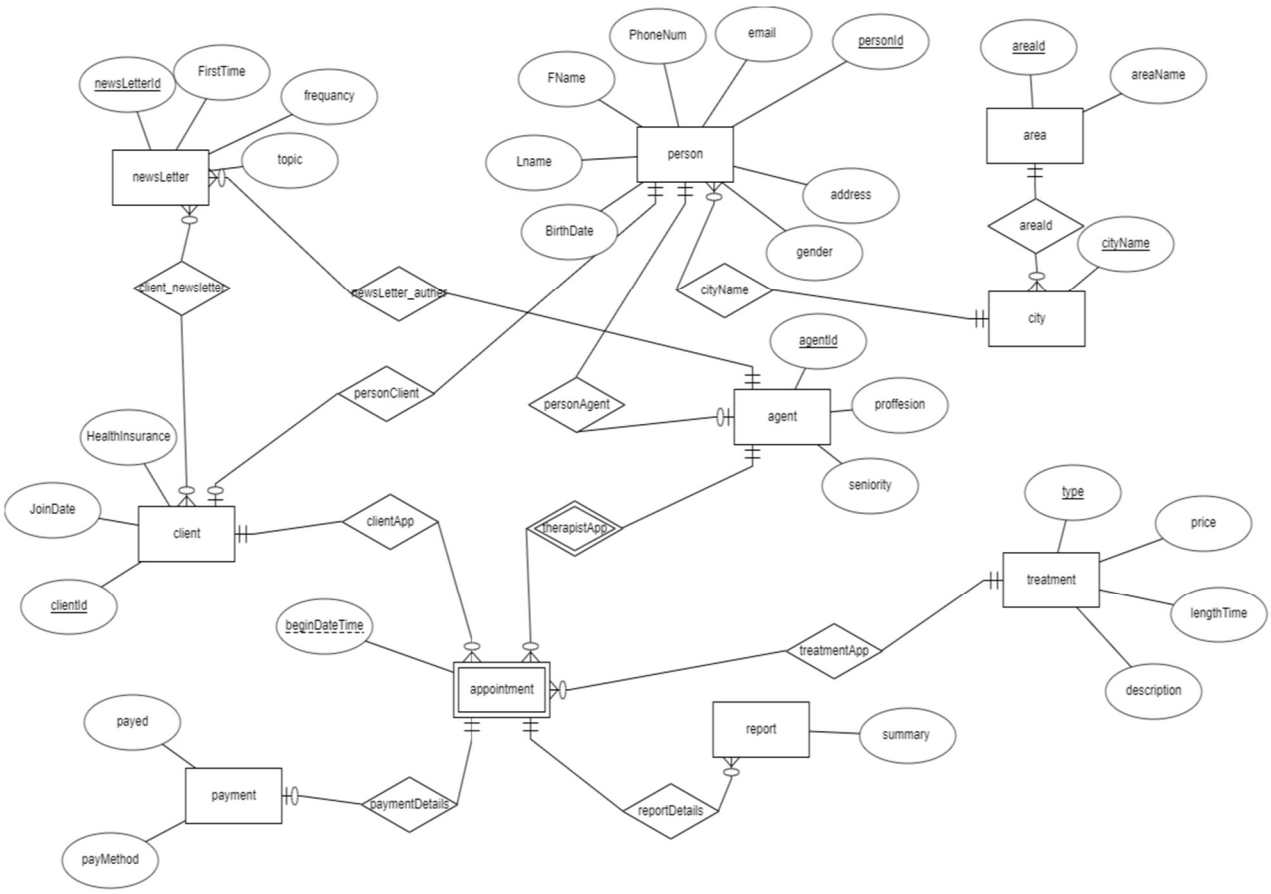
Client_newsletter that has all the newsletters a client is subscribed to.

Will have these attributes:

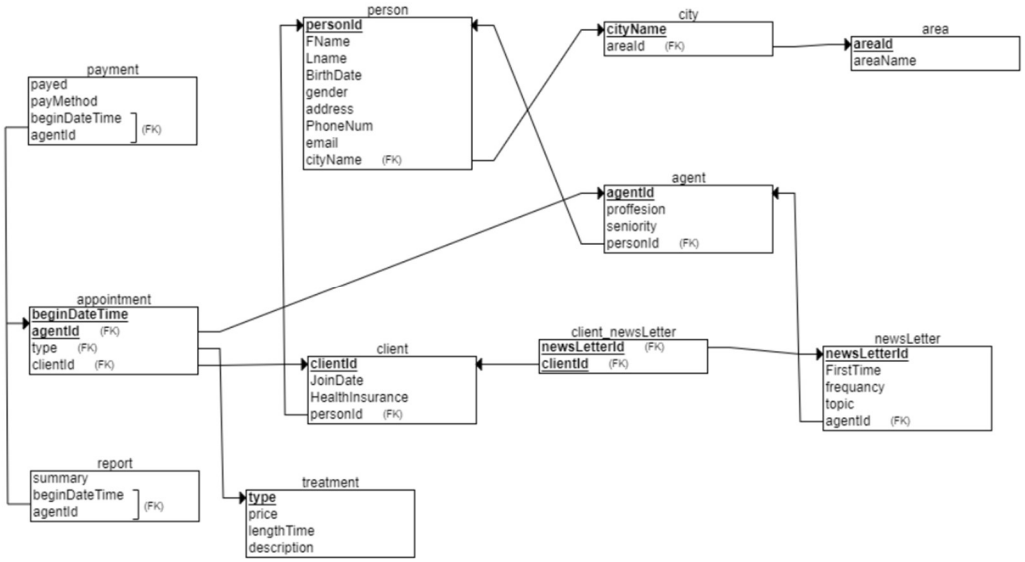
Clientid – the client that is subscribed

Newsletterid – the id of the newsletter.

The new erd:


PhoneNum
email
personId
areaId
areaName
FirstTime
newsLetterId
frequancy
FName
person
area
topic
newsLetter
Lname
address
BirthDate
gender
areaId
cityName
client_newsletter
newsLetter_author
cityName
city
agentId
personClient
personAgent
agent
proffesion
HealthInsurance
seniority
type
JoinDate
client
clientApp
therapistApp
price
treatment
clientId
beginDateTime
lengthTime
treatmentApp
appointment
description
payed
report
summary
payment
paymentDetails
reportDetails
payMethod

Dsd:


person
personId
FName
Lname
BirthDate
gender
address
PhoneNum
email
cityName (FK)
city
cityName
areaId (FK)
area
areaId
areaName
payment
payed
payMethod
beginDateTime
agentId (FK)
agent
agentId
proffesion
seniority
personId (FK)
appointment
beginDateTime
agentId (FK)
type (FK)
clientId (FK)
client
clientId
JoinDate
HealthInsurance
personId (FK)
client_newsLetter
newsLetterId (FK)
clientId (FK)
newsLetter
newsLetterId
FirstTime
frequancy
topic
agentId (FK)
report
summary
beginDateTime
agentId (FK)
treatment
type
price
lengthTime
description

Sql scripts for the new entities:

For newsletter:

```
CREATE TABLE newsLetter
(
  newsLetterId INT NOT NULL,
  FirstTime DATE NOT NULL,
  frequancy INT NOT NULL,
  topic VARCHAR(100) NOT NULL,
  agentId INT NOT NULL,
  PRIMARY KEY (newsLetterId),
  FOREIGN KEY (agentId) REFERENCES agent(agentId)
);
```

For client_newsletter:

```
CREATE TABLE client_newsLetter
(
  newsLetterId INT NOT NULL,
  clientId INT NOT NULL,
  PRIMARY KEY (newsLetterId, clientId),
  FOREIGN KEY (newsLetterId) REFERENCES newsLetter(newsLetterId),
  FOREIGN KEY (clientId) REFERENCES client(clientId)
);
```

The sql from data generator:

```
insert into SYSTEM.NEWSLETTER (NEWSLETTERID, FIRSTTIME, FREQUANCY, TOPIC,
AGENTID)
values (2000, to_date('05-03-2005', 'dd-mm-yyyy'), 1, 'Anxiety Coach', 1001);

insert into SYSTEM.NEWSLETTER (NEWSLETTERID, FIRSTTIME, FREQUANCY, TOPIC,
AGENTID)
values (2001, to_date('26-01-2005', 'dd-mm-yyyy'), 1, 'A Storied Mind',
1000);

insert into SYSTEM.NEWSLETTER (NEWSLETTERID, FIRSTTIME, FREQUANCY, TOPIC,
AGENTID)
values (2002, to_date('14-06-2005', 'dd-mm-yyyy'), 3, 'Bipolar Burble Blog',
1001);

insert into SYSTEM.NEWSLETTER (NEWSLETTERID, FIRSTTIME, FREQUANCY, TOPIC,
AGENTID)
values (2003, to_date('09-04-2005', 'dd-mm-yyyy'), 2, 'The Secret Life of a
Manic Depressive', 1000);

insert into SYSTEM.NEWSLETTER (NEWSLETTERID, FIRSTTIME, FREQUANCY, TOPIC,
AGENTID)
values (2004, to_date('26-01-2005', 'dd-mm-yyyy'), 2, 'A Storied Mind',
1002);
```

```
insert into SYSTEM.NEWSLETTER (NEWSLETTERID, FIRSTTIME, FREQUANCY, TOPIC,
AGENTID)
values (2005, to_date('25-04-2005', 'dd-mm-yyyy'), 3, 'Bipolar Burble Blog',
1003);

insert into SYSTEM.NEWSLETTER (NEWSLETTERID, FIRSTTIME, FREQUANCY, TOPIC,
AGENTID)
values (2006, to_date('20-01-2005', 'dd-mm-yyyy'), 3, 'Anxiety Slayers',
1000);

insert into SYSTEM.NEWSLETTER (NEWSLETTERID, FIRSTTIME, FREQUANCY, TOPIC,
AGENTID)
values (2007, to_date('20-04-2005', 'dd-mm-yyyy'), 1, 'Anxiety Coach', 1002);

insert into SYSTEM.NEWSLETTER (NEWSLETTERID, FIRSTTIME, FREQUANCY, TOPIC,
AGENTID)
values (2008, to_date('19-10-2005', 'dd-mm-yyyy'), 3, 'The Secret Life of a
Manic Depressive', 1001);

insert into SYSTEM.NEWSLETTER (NEWSLETTERID, FIRSTTIME, FREQUANCY, TOPIC,
AGENTID)
values (2009, to_date('03-02-2005', 'dd-mm-yyyy'), 1, 'A Storied Mind',
1003);

insert into SYSTEM.NEWSLETTER (NEWSLETTERID, FIRSTTIME, FREQUANCY, TOPIC,
AGENTID)
values (2010, to_date('31-01-2005', 'dd-mm-yyyy'), 1, 'Bipolar Burble Blog',
1001);

insert into SYSTEM.NEWSLETTER (NEWSLETTERID, FIRSTTIME, FREQUANCY, TOPIC,
AGENTID)
values (2011, to_date('30-06-2005', 'dd-mm-yyyy'), 2, 'The Secret Life of a
Manic Depressive', 1000);

insert into SYSTEM.NEWSLETTER (NEWSLETTERID, FIRSTTIME, FREQUANCY, TOPIC,
AGENTID)
values (2012, to_date('28-11-2005', 'dd-mm-yyyy'), 1, 'Anxiety Slayers',
1001);

commit;
```

Queries

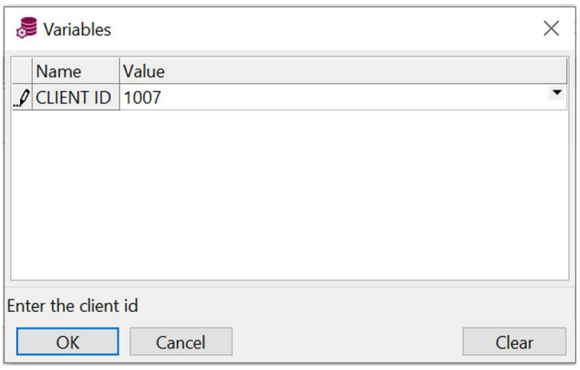
1.

Clients can have sessions with several agents.

We wanted to find for a specific client if there are any newsletters from a therapist, where he had sessions with and didn't subscribe to his/her newsletter, in order to subscribe them.

```
select *
from
(select newsletterid, clientid
from newsletter natural join appointment
where appointment.clientid = &<name="CLIENT ID" hint="Enter the client id"> )
minus
(select newsletterid, clientid from client_newsletter where clientid =
&<name="CLIENT ID" hint="Enter the client id">);
```

Example:



Result:

These are the newsletter id's that he isn't subscribed to and we will offer him to subscribe to.

	NEWSLETTERID	CLIENTID
1	2000	1007
2	2001	1007
3	2003	1007
4	2008	1007
5	2009	1007

2.

By the end of the day we need appointments to get income. The newsletter can give us an indication of the effect it has on the clients and whether it can translate into payed appointments. We wanted to get some statistics for each client to see if the number of newsletters they are subscribed to, influences the number of appointments. The next procedure shows the number of subscription and appointments for every client.

If the conversion rate (number of appointments / number of newsletters) is larger than 1 , it's considered 'high' else, it's 'low'.

Obviously this is an example and it can be upgraded a lot to get some REAL statistics.

```
create or replace procedure check_conversion_newsletter
is

c_id number;
sum_newsletter number;
sum_appointments number;
conversion_rate number;

cursor appointments_newsletter is
select clientid from client;
BEGIN
OPEN appointments_newsletter;
LOOP
 FETCH appointments_newsletter into c_id;
 EXIT WHEN appointments_newsletter%notfound;
 sum_newsletter:=calc_num_of_newsletter(c_id);
 sum_appointments:=calc_num_of_appointments(c_id);
 dbms_output.put_line('client: '||c_id|| ' appointments:
'||sum_appointments||
 ' ------ subscriptions: '|| sum_newsletter);
 conversion_rate:=sum_appointments/sum_newsletter;
 IF conversion_rate > 1
   then dbms_output.put_line('HIGH');
   else
     dbms_output.put_line('Low');
```

```
   END IF;
dbms_output.put_line('conversion: '|| conversion_rate);
   dbms_output.put_line('**************');
END LOOP;
CLOSE appointments_newsletter;

END check_conversion_newsletter;
```

This uses two other functions calc_num_of_newsletter, calc_num_of_appointments that we wrote to calculated the number of newsletters and appointments for each client. We use that info for the procedure.

Example:

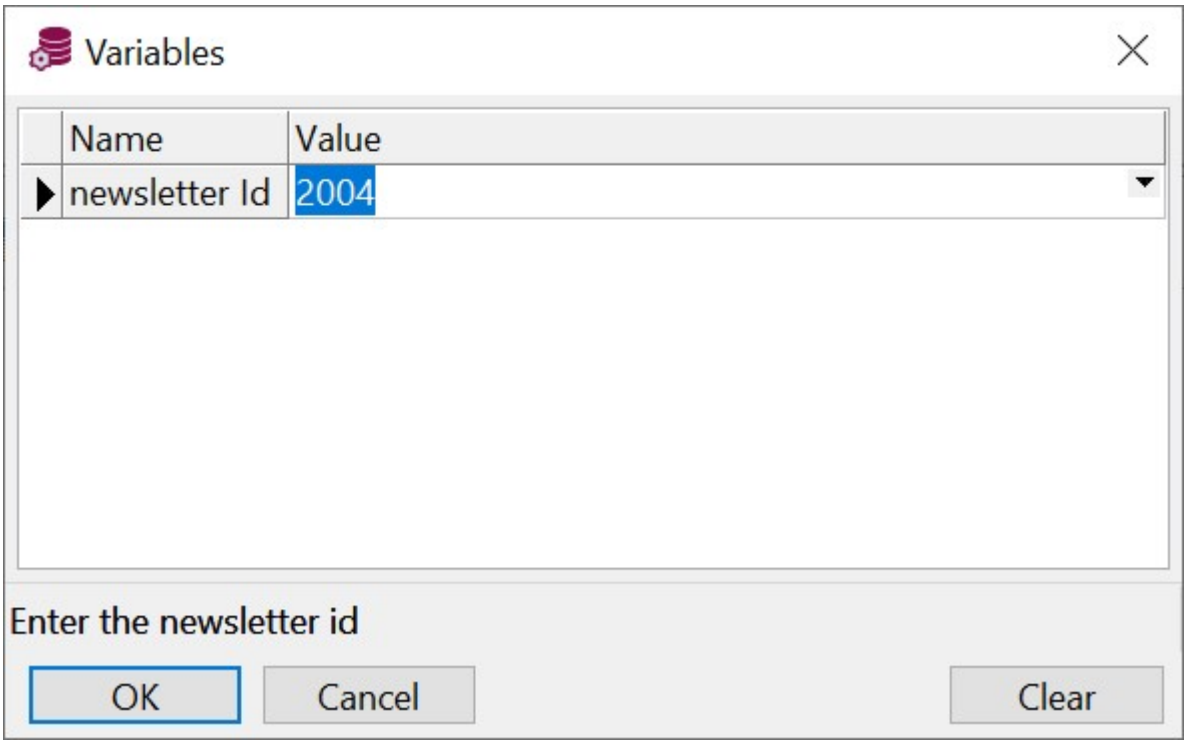
When running the procedure this is the output:

```
client: 1000 appointments: 4 ------ subscriptions: 2
HIGH
conversion: 2
**************
client: 1001 appointments: 3 ------ subscriptions: 1
HIGH
conversion: 3
**************
client: 1002 appointments: 3 ------ subscriptions: 4
Low
conversion: .75
**************
client: 1003 appointments: 3 ------ subscriptions: 6
Low
conversion: .5
**************
client: 1004 appointments: 0 ------ subscriptions: 2
Low
conversion: 0
**************
client: 1007 appointments: 3 ------ subscriptions: 3
Low
conversion: 1
**************
```

Query to give a list of all the email addresses for a specific newsletter. It is needed in order to know to what list to send:

```
select email_
from client, client_newsletter, person
where newsletterid = &<name="newsletter Id" hint="Enter the newsletter id">
```

```
and client.clientid=client_newsletter.clientid and person.personid =
client.personid
```



Result:

The newsletter needs to be sent to these addresses:

	EMAIL_
1	mike.c@taycorfinancial.br
2	milla.p@bigdoughcom.at

Views

List of newsletter that we want to subscribe a client (the client has sessions with the authors of these newsletters)

```
create view as add_newsletter_for_client
select *
from
(select newsletterid, clientid
from newsletter natural join appointment
where appointment.clientid = 1002)
minus
(select newsletterid, clientid from client_newsletter where clientid = 1002);
```

View for the query to give a list of all the email addresses for a specific newsletter:

```
create view email_newsletter_list as
select email_
from client, client_newsletter, person
where newsletterid = 2002
and client.clientid=client_newsletter.clientid and person.personid =
client.personid
```

good to know: Interestingly You can't pass a parameter to a view.
We read about some complex options online to overpass this, but we didn’t implement it in this project.

סיכום

כל הפרוייקט נעשה בגישה מעשית, כאשר את ההשראה קיבלנו מחזקי שאשתו מנהלת קליניקה של טיפולים רגשיים, והמטרה היתה שבמידה ונרצה נוכל להשמיש את הפרוייקט בקלות יחסית לניהול הקליניקה.

באופן אישי הקורס לימד אותנו המון,
החל מERD PLUS שהוא כלי מצויין, שבנוסף לזה שהוא ממחיש באופן ויזואלי את מבנה בסיס הנתונים, הוא גם יוצר אוטומטית את קוד הsql של ה"create tables" שלאחר תיקונים קלים כבר יכול לשמש לבניה של טבלאות הORACLE SQL.
לאורך כל הפרוייקט, גם בשלבים המתקדמים יותר כשרצינו לבנות שאילתה/פונקציה/פרוצדורה כלשהי נעזרנו בו כדי להבין איך לכתוב את השאילתה וממה היא אמורה להיות מורכבת.
וגם בעתיד כאשר נרצה לבנות בסיס נתונים כלשהו אין לנו ספק שנעזר בו.

מהפרוייקט למדנו בעיקר כיצד לבנות בסיס נתונים בצורה נכונה, כך שלא יהיה חוסר/כפלויות בנתונים, וגם להקל בהמשך על יצירת השאילתות.

לבד מזאת למדנו כיצד לבנות שאילתות מעשיות בצורה נכונה תוך שימוש בסינטקס של שפת SQL ספציפית (Oracle SQL במקרה זה).

גם למדנו להכיר ולשלב בפרוייקט את מגוון כלי העזר שמציע בסיס נתונים מודרני (לדוגמא פרוצדורות, טריגרים וכו'), וכן איך להרחיב בסיס נתונים קיים לאזור חדש.
תקופת הקורונה כמובן איתגרה ודחקה אותנו לשפר מספר מיומנויות נוספות (עבודה בצוות מרחוק, סינכרון העבודה בענן בין חברי הצוות, עבודה עם תמיכה מינימלית מצוות הקורס עקב המרחק הפיזי, ועוד ועוד)
למרות זאת, אנחנו חושבים שסיימנו את הפרוייקט בהצלחה. (ברגע אפשר די בקלות להשמיש את הפרוייקט לניהול הקליניקה של משפחת באטוויניק).
והצלחנו להפיק המון מהקורס למרות הקשיים.
הרבה מאוד בזכות המרצה מר אריה ויזן

תודה רבה,
יעקב פרדמן וחזקי באטוויניק.