COMP 421: Project Pt. 2

Edoardo Tarek Holzl (260745790), Marin Thiercelin (260746335), William Burgess (260477209), Clara Kang (260558716)

1. Updated Relational Schema

Entities:
Client(Cid, Name, streetNum,street, city, country Creationdate)
Employee(Eid, StDate, Salary, Bid,workingDays, startHour, endHour) (Bid ref Branch)
Branch(Bid, StreetNumber, Street, City, Country, OpeningTime,ClosingTime)
Product(PrId, Brand, Name, Type, Year, Available, Bid) (Bid ref Branch)
ForRent(PrId, Condition) (Prid ref Product)
ForSale(PrId, Price) (PrId ref Product)
Salesman(Eid) (Eid ref Employee)
Manager(Eid,Bid) (Eid ref Employee, Bid ref Branch)
Fee(Fid,Price, Duration)
Payment(PyId, Discnt, Date, Mthod, Amt, Eid,Cid) (eid ref Employee, cid ref Employee)

Relationships:
Rents(Cid,PyId, PrId, InitCndit, EndDate, StartDate) (PyId ref Payment, PrId ref Product)
Buys(PrId, PyId) (PrId ref Product, PyId ref Payment)
Rates(rateid , eID, cid, Ratings) (Cid ref Client, Eid ref Employee)
PaysFor(prID, fID) (prID ref ForRent, fID ref Fee)

2. Create Statements

```
/* Entities */

CREATE TYPE ProductType AS ENUM('Ski', 'Snowboard', 'Poles', 'SkiBoots', 'Snowboots',
'Helmets', 'Skiwear', 'Accessories');

CREATE TYPE PymtMethod AS ENUM ( 'cash', 'credit', 'debit');


CREATE TYPE RentingDuration AS ENUM ('1_HOUR', '1_DAY', '2_DAYS', '1_WEEK',
'1_MONTH','1_YEAR');

CREATE TYPE PrConditionType AS ENUM ( 'Good', 'Bad', 'Medium');

CREATE TABLE Client(
       cid INTEGER PRIMARY KEY,
       cName VARCHAR(30)  NOT NULL,
       streetNum INTEGER,
       street VARCHAR(30),
       city VARCHAR(30),
       country VARCHAR(20),
       creationDate DATE DEFAULT CURRENT_DATE
);


CREATE TABLE Branch(
       Bid INTEGER PRIMARY KEY,
       StreetNumber INTEGER,
       Street VARCHAR(30),
       City VARCHAR(30),
       Country VARCHAR(20),
       OpeningTime TIME,
       ClosingTime TIME
);


CREATE TABLE Employee(
       eid INTEGER PRIMARY KEY,
       eName VARCHAR(30) NOT NULL,
       startDate DATE DEFAULT CURRENT_DATE,
       salary INTEGER NOT NULL, CHECK (salary > 0),
       bid INTEGER,
       workingDays CHAR(7),
       CONSTRAINT a CHECK (
              workingDays SIMILAR TO '[1,\_][2,\_][3,\_][4,\_][5,\_][6,\_][0,\_]'
       ),
       startTime TIME,
       endTime TIME,
       FOREIGN KEY(Bid) REFERENCES Branch ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE Product (
       PrId INTEGER PRIMARY KEY,
       Brand VARCHAR(20) NOT NULL,
       pName VARCHAR(20),
       pType ProductType NOT NULL,
       pYear INTEGER,
       Available BOOLEAN NOT NULL,
       Bid INTEGER,
       FOREIGN KEY(Bid) REFERENCES Branch ON DELETE CASCADE ON UPDATE CASCADE
);



CREATE TABLE ForRent (
       PrId INTEGER PRIMARY KEY,
       prCondition PrConditionType NOT NULL,
       FOREIGN KEY(PrId) REFERENCES Product ON DELETE CASCADE ON UPDATE CASCADE
);


CREATE TABLE ForSale (
       prID INTEGER PRIMARY KEY,
       Price INTEGER NOT NULL CHECK(Price >= 0),
       FOREIGN KEY(prID) REFERENCES Product ON DELETE CASCADE ON UPDATE CASCADE
);


CREATE TABLE Salesman(
       Eid INTEGER PRIMARY KEY,
       FOREIGN KEY(Eid) REFERENCES Employee ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Manager (
    Eid INTEGER PRIMARY KEY,
    Bid INTEGER NOT NULL,
    FOREIGN KEY( Bid ) REFERENCES Branch ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY( Eid ) REFERENCES Employee ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Fee (
       Fid INTEGER PRIMARY KEY,
       Price INTEGER,
       Duration RentingDuration
);

CREATE TABLE Payment(
       PyId INTEGER PRIMARY KEY,
       Discnt INTEGER CHECK (Discnt >= 0 AND Discnt <= 100),
       pyDate TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP(2),
       Method PymtMethod NOT NULL,
       Amount REAL CHECK ( Amount >= 0 ) NOT NULL,
       Eid INTEGER,
       Cid INTEGER ,
       FOREIGN KEY(Eid) REFERENCES Employee ON DELETE SET NULL ON UPDATE CASCADE ,
       FOREIGN KEY(Cid) REFERENCES Client ON DELETE SET NULL ON UPDATE CASCADE
);
```

/* Relationships */

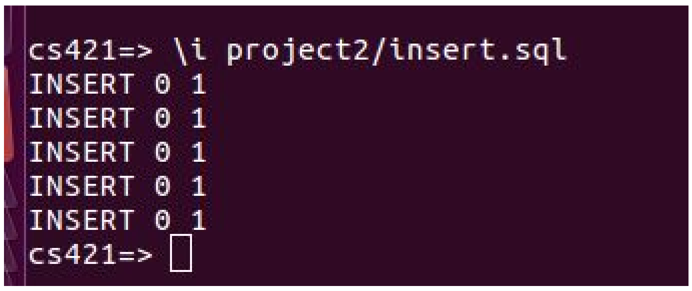
```
CREATE TABLE RENTS(
        CONSTRAINT RentId   PRIMARY KEY(Cid,PyId,PrId),
        Cid    INTEGER      NOT NULL REFERENCES Client(Cid) ON DELETE CASCADE ON UPDATE
CASCADE,
        PyId   INTEGER      NOT NULL REFERENCES Payment(PyId) ON DELETE CASCADE ON UPDATE
CASCADE,
        PrId   INTEGER      NOT NULL REFERENCES ForRent(PrId) ON DELETE CASCADE ON UPDATE
CASCADE,

        InitCndit   VARCHAR(50)    NOT NULL,
        StartDate   TIMESTAMP   NOT NULL  DEFAULT   CURRENT_TIMESTAMP(2),
        EndDate      TIMESTAMP   NOT NULL

);

CREATE TABLE BUYS(
        CONSTRAINT BuyId PRIMARY KEY(PrId, PyId),

        PrId INTEGER NOT NULL REFERENCES ForSale(PrId) ON DELETE CASCADE ON UPDATE CASCADE,
        PyId INTEGER REFERENCES Payment(PyId) ON DELETE SET NULL ON UPDATE CASCADE
);

CREATE TABLE RATES(
        CONSTRAINT RateId PRIMARY KEY(RateId,Eid),
        RateId INTEGER NOT NULL,
        Cid INTEGER REFERENCES Client(Cid) ON DELETE SET NULL ON UPDATE CASCADE,
        Eid INTEGER NOT NULL REFERENCES Employee(Eid) ON DELETE CASCADE ON UPDATE CASCADE,
        Rating INTEGER NULL        CHECK(Rating >= 1 and Rating <= 5)
);

CREATE TABLE PAYSFOR(
        CONSTRAINT PaysForId        PRIMARY KEY(PrId,Fid),
        PrId INTEGER  NOT NULL REFERENCES ForRent(PrId) ON DELETE CASCADE ON UPDATE CASCADE,
        Fid INTEGER   NOT NULL REFERENCES Fee(Fid) ON DELETE CASCADE ON UPDATE CASCADE
);
```

3. Insertions

Statements :

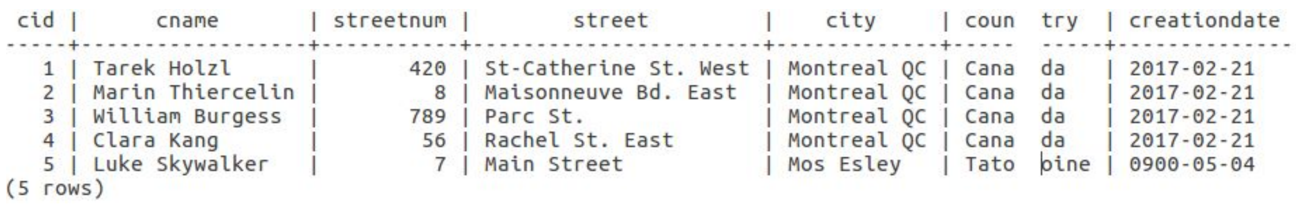
```
INSERT INTO Client VALUES (1, 'Tarek Holzl', 420, 'St-Catherine St. West','Montreal
QC','Canada', CURRENT_DATE);
INSERT INTO Client VALUES (2, 'Marin Thiercelin', 8, 'Maisonneuve Bd. East','Montreal
QC','Canada', CURRENT_DATE);
INSERT INTO Client VALUES (3, 'William Burgess', 789, 'Parc St.','Montreal QC','Canada',
CURRENT_DATE);
INSERT INTO Client VALUES (4, 'Clara Kang', 56, 'Rachel St. East','Montreal QC','Canada',
CURRENT_DATE);
INSERT INTO Client VALUES (5, 'Luke Skywalker', 7, 'Main Street','Mos Esley','Tatooine',
'0900-05-04');
```

Output :



```
cs421=> \i project2/insert.sql
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
cs421=> 
```

Table description :



```
 cid |      cname       | streetnum |        street         |     city     | coun  try   | creationdate
-----+------------------+-----------+-----------------------+--------------+-----  -----+--------------
   1 | Tarek Holzl      |       420 | St-Catherine St. West | Montreal QC  | Cana  da    | 2017-02-21
   2 | Marin Thiercelin |         8 | Maisonneuve Bd. East  | Montreal QC  | Cana  da    | 2017-02-21
   3 | William Burgess  |       789 | Parc St.              | Montreal QC  | Cana  da    | 2017-02-21
   4 | Clara Kang       |        56 | Rachel St. East       | Montreal QC  | Cana  da    | 2017-02-21
   5 | Luke Skywalker   |         7 | Main Street           | Mos Esley    | Tato  oine  | 0900-05-04
(5 rows)
```

4. Data Generation

Data was inserted in the database using an Automated program that creates INSERT statements in Scala programming language. The program generated 5 branches, 80 employees, 100 clients, 250 products and 70 payments

Samples:

Branches

```
bid | streetnumber   |   street             |   city           | country | openingtime | closingtime
-----+-------------------+-------------------+-----------------+----------+------------------+-------------
  10 |             4316 | Hill street         | Dubai            | UAE   | 07:30:00        | 16:00:00
   9 |              800 | Hill street         | Dubai            | China   | 07:30:00      | 17:00:00
   7 |             6712 | Eagle street       | Jeddah          | Canada  | 08:00:00    | 16:00:00
   6 |             3205 | Phairview          | Jeddah          | France  | 07:30:00     | 16:00:00
   5 |             5679 | Phairview          | Dubai            | Canada  | 08:00:00    | 18:00:00
   4 |             1669 | Amherst west   | Dubai            | UAE   | 09:00:00        | 18:00:00
   3 |               90 | Sun Street          | Hong Kong    | China   | 07:30:00      | 18:00:00
   2 |             7262 | Strawberry Lane | Beijing        | France  | 09:00:00     | 16:00:00
```

Clients

```
  cid | cname                          | streetnum |    street         |   city     | country | creationdate
-----+------------------------------ +-------------+----------------+-----------+----------+-------------
   1 | Kit Robinson              |          5397 | Hill street         | Toronto    | India   | 2015-10-15
   2 | Reginald Rodriguez     |          702 | Strawberry Lane | Montreal  | UAE       | 2016-03-13
   3 | Delfina Adams             |          1707 | Sunset Avenue   | Toronto   | UAE      | 2011-11-09
   4 | Katerine Young           |          5256 | Anne street      | Jeddah         | UAE   | 2015-07-26
   5 | Jettie Turner               |          4538 | Amherst west   | Hong Kong | Canada  | 2009-11-15
   6 | Janette Perez              |          5459 | Anne street      | Toronto    | India   | 2011-10-22
   7 | Vera Carter                 |          9998 | Anne street      | Hong Kong | China    | 2014-11-17
   8 | Lucie Turner                |          2635 | Anne street      | New Delhi | China   | 2013-07-22
   9 | Enola Parker               |          7845 | Sunset Avenue   | New Delhi | France  | 2009-08-25
  10 | Vera Green                  |          6652 | Anne street      | Toronto    | Turkey  | 2010-12-14
```

Employees

```
 eid |    ename                       | startdate     | salary | bid | workingdays | starttime | endtime
-----+------------------------------+---------------+---------+-------------+---------+-----------+------------
   2 | Melinda Allen              | 2007-04-08 |   2846 |    1 | 1_3_560  | 09:00:00  | 17:00:00
   3 | Keri Mitchell                | 2007-12-29 |   2632 |    1 | 123_56_  | 07:30:00  | 16:00:00
   4 | Kazuko Baker               | 2007-06-29 |   2885 |    1 | 1_3_560  | 07:30:00  | 16:00:00
   5 | Jess Scott                    | 2015-03-15 |   2937 |    1 | 1_3_560  | 08:00:00  | 18:00:00
   7 | Tameika Anderson        | 2011-04-16 |   2309 |    1 | _234_60  | 08:00:00  | 16:00:00
```

```
  8 | Tameika Carter            | 2016-07-29 |   2917 |   1 | 1_3_560  | 08:00:00  | 16:00:00
  9 | Cheri Jackson     | 2008-09-10 |   2792 |   1 | _234_60  | 09:00:00  | 18:00:00
 10 | Melinda Robinson | 2011-12-30 |   2733 | 1 | 123__60 | 09:00:00  | 18:00:00
```

Managers

```
  eid | bid
-----+-----
  22 |   2
  43 |   3
  64 |   4
  85 |   5
 127 |   7
 148 |   8
 169 |   9
 190 |  10
   1 |   8
 106 |   9
```

Salesman

```
 eid
-----
   2
   3
   4
   5
   6
   7
   8
   9
  10
```

Products

```
 prid |   brand   | pname |       ptype   | pyear | available | bid
------+-----------+-------+-------------+-------+-----------+-----
1 | Armada      |          | Helmets       |  2011 | t        |   1
2 | Armada      |          | Helmets       |  2012 | t        |   1
3 | Rossignol  |          | Poles           |  2008 | t        |   1
4 | Salomon    |          | SkiBoots      |  2009 | t        |   1
5 | Salomon    |          | Poles           |  2010 | t        |   1
6 | Armada     |          | Snowboard   |  2012 | t        |   1
7 | Salomon    |          | Helmets       |  2010 | t        |   1
8 | Burton       |          | Snowboots   |  2012 | t        |   1
9 | Rossignol   |          | Snowboard   |  2009 | t        |   1
10 | Armada    |          | SkiBoots      |  2008 | t        |   1
```

ForSale

```
 prid   | price
--------+-------
1 |       136
2 |        82
3 |        69
4 |        79
5 |        56
6 |        493
7 |        88
8 |        212
9 |        500
```

ForRent

```
prid  | prcondition
------+-------------
  201 | Medium
  202 | Medium
  203 | Bad
  204 | Medium
  205 | Good
  206 | Good
  207 | Medium
  208 | Bad
```

Payment

```
pyid | discnt |        pydate        | method | amount | eid | cid
------+--------+---------------------+--------+--------+-----+-----
1 |       14 | 2012-03-14 11:25:00 | credit |   1050 |  46 |   4
3 |       45 | 2013-09-05 09:47:00 | credit |    281 |  22 | 246
4 |       25 | 2013-02-16 10:04:00 | debit  |     23 |  50 | 129
5 |       22 | 2010-05-14 12:40:00 | cash   |    100 | 158 | 149
7 |       21 | 2013-11-02 16:29:00 | credit |    156 |  40 | 135
8 |       47 | 2009-07-11 10:37:00 | credit |    200 |  31 |  32
9 |       35 | 2014-03-19 15:45:00 | credit |    399 |   1 | 134
```

Rents

```
cid | pyid | prid | initcndit        |         startdate          |          enddate
-----+------+------+-----------------+---------------------------+---------------------------
 246 |    3 |  227 | Medium          | 2013-09-05 09:47:00 | 2013-09-05 10:47:00
 246 |    3 |  229 | Medium          | 2013-09-05 09:47:00 | 2014-09-05 09:47:00
 246 |    3 |  221 | Medium          | 2013-09-05 09:47:00 | 2013-09-06 09:47:00
 129 |    4 |  255 | Medium          | 2013-02-16 10:04:00 | 2013-02-16 11:04:00
 129 |    4 |  250 | Bad             | 2013-02-16 10:04:00 | 2013-02-23 10:04:00
 149 |    5 |  354 | Medium          | 2010-05-14 12:40:00 | 2011-05-14 12:40:00
 149 |    5 |  358 | Medium          | 2010-05-14 12:40:00 | 2010-05-21 12:40:00
 130 |    6 |  264 | Medium          | 2015-07-23 14:08:00 | 2015-08-23 14:08:00
 135 |    7 |  238 | Bad             | 2013-11-02 16:29:00 | 2013-12-02 16:29:00
```

Buys

```
 prid | pyid
------+------
   60 |    1
   54 |    1
   41 |    1
   90 |    2
   34 |    3
   73 |    6
   24 |    7
   18 |    9
    8 |    9
```

PaysFor

```
 prid | fid
------+-----
  201 |   6
  201 |   7
  201 |  18
  201 |  24
  201 |  27
  202 |   6
  202 |  11
```

Fee

```
fid | price | duration
-----+-------+----------
   1 |   450 | 1_YEAR
   2 |   400 | 1_YEAR
   3 |   350 | 1_YEAR
   4 |   100 | 1_YEAR
   5 |    70 | 1_YEAR
```

```
  6 |    50 | 1_YEAR
  7 |   250 | 1_MONTH
  8 |   200 | 1_MONTH
```

```
Rates
rateid | cid | eid | rating
--------+-----+-----+--------
  1     |   4 |  46 |    1
  2     | 251 |  95 |    5
  3     | 246 |  22 |    5
  4     | 129 |  50 |    4
  5     | 149 | 158 |    2
  9     | 134 |   1 |    4
 11     |  41 |  77 |    4
 13     |  48 | 150 |    4
 14     |  44 | 173 |    5
```

5. Queries

The relations used in the queries are:

1. RENTS

```
CREATE TABLE RENTS(
    CONSTRAINT RentId   PRIMARY KEY(Cid,PyId,PrId),
    Cid INTEGER NOT NULL REFERENCES Client(Cid) ON DELETE CASCADE ON UPDATE CASCADE,
    PyId INTEGER NOT NULL REFERENCES Payment(PyId) ON DELETE CASCADE ON UPDATE CASCADE,
    PrId INTEGER NOT NULL REFERENCES ForRent(PrId) ON DELETE CASCADE ON UPDATE CASCADE,
    InitCndit   VARCHAR(50) NOT NULL,
    StartDate   TIMESTAMP   NOT NULL    DEFAULT   CURRENT_TIMESTAMP(2),
    EndDate     TIMESTAMP   NOT NULL
);
```

2. Product

```
CREATE TABLE Product (
    PrId INTEGER PRIMARY KEY,
    Brand VARCHAR(20) NOT NULL,
    pName VARCHAR(20),
    pType ProductType NOT NULL,
    pYear INTEGER,
    Available BOOLEAN NOT NULL,
    Bid INTEGER,
    FOREIGN KEY(Bid) REFERENCES Branch ON DELETE CASCADE ON UPDATE CASCADE
);
```

3. BUYS

```
CREATE TABLE BUYS(
    CONSTRAINT  BuyId PRIMARY KEY(PrId, PyId),
    PrId INTEGER NOT NULL REFERENCES ForSale(PrId) ON DELETE CASCADE ON UPDATE CASCADE,
    PyId INTEGER REFERENCES Payment(PyId) ON DELETE SET NULL ON UPDATE CASCADE

);
```

4. Payment

```
CREATE TABLE Payment(
    PyId INTEGER PRIMARY KEY,
    Discnt INTEGER CHECK (Discnt >= 0 AND Discnt <= 100),
    pyDate TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP(2),
    Method PymtMethod NOT NULL,
    Amount REAL CHECK ( Amount >= 0 ) NOT NULL,
    Eid INTEGER,
    Cid INTEGER ,
    FOREIGN KEY(Eid) REFERENCES Employee ON DELETE SET NULL ON UPDATE CASCADE ,
    FOREIGN KEY(Cid) REFERENCES Client ON DELETE SET NULL ON UPDATE CASCADE
);
```

5. Client

```
CREATE TABLE Client(
    cid INTEGER PRIMARY KEY,
    cName VARCHAR(30)  NOT NULL,
    streetNum INTEGER,
    street VARCHAR(30),
    city VARCHAR(30),
    country VARCHAR(20),
    creationDate DATE DEFAULT CURRENT_DATE
);
```

6. ForRent

```
CREATE TABLE ForRent (
    PrId INTEGER PRIMARY KEY,
    prCondition PrConditionType NOT NULL,
    FOREIGN KEY(PrId) REFERENCES Product ON DELETE CASCADE ON UPDATE CASCADE
);
```

Query #1: Get the types and the brands of the products that are overdue (assuming today is 2008-01-01), the customers that hold them, and the date they are supposed to return them.

```
SELECT p.Brand, p.pType, c.cName, r.EndDate
FROM RENTS r JOIN Product p
ON (r.PrId = p.PrId AND r.EndDate > '2008-01-01')
JOIN Client c
ON (r.Cid = c.Cid);
```

Query #2: Get the countries of customers who have spent $500 buying products at the stores, show the number of people of each country group, and rank them according to this number

```
CREATE VIEW CustmrWithAmt(Cid, amt)
AS SELECT p.Cid AS Cid, SUM(p.Amount) AS amt
FROM BUYS b, Payment p
WHERE  b.PyId = p.PyId
GROUP BY p.Cid;

SELECT c.country, COUNT(*) AS num_people
FROM CustmrWithAmt cm, Client c
WHERE cm.Cid = c.Cid AND cm.amt > 500
GROUP BY c.country
ORDER BY num_people DESC;
```

Query #3: Get the types of the 'forRent' products where over 8 of the products are of condition "Bad"

```
SELECT p.pType, COUNT(*) AS numOfBadProduct
FROM ForRent f, Product p
WHERE f.PrId = p.PrId AND f.prCondition = 'Bad'
GROUP BY p.pType
HAVING COUNT(*) >= 8;
```

Query #4: Get the brands of products which the total revenue is over 1000 in the year 2016, rank them according to revenue, and show revenue.

```
SELECT pr.Brand, SUM(py.Amount) as amt
FROM BUYS b JOIN Product pr
ON b.PrId = pr.PrId
JOIN Payment py
ON b.PyId = py.PyId AND to_char(py.pyDate, 'YYYY') = '2015'
GROUP BY pr.Brand
HAVING SUM(py.Amount) > 1000
ORDER BY amt;
```

Query #5: For each type of 'forRent' product, get the age(year manufactured) of the majority

```
CREATE VIEW ProductYear(pType, year, num)
AS SELECT p.pType, p.pYear, COUNT(*) AS num
FROM ForRent f, Product p
WHERE f.PrId = p.PrId
GROUP BY p.pType, p.pYear;

CREATE VIEW ProductYearMax(pType, num)
AS SELECT p.pType, MAX(p.num)
FROM ProductYear p
GROUP BY p.pType;

SELECT p1.pType, p1.year as majorAge
FROM ProductYear p1, ProductYearMax p2
WHERE p1.pType = p2.pType AND p1.num = p2.num;
```

Script:

```
cs421g21@comp421:~$ psql cs421 < query.sql > results.log
Password:
cs421g21@comp421:~$ cat results.log
 brand            |   ptype        |      cname            | enddate
------------------+----------------+------------------+----------------------------------
 Armada           | SkiBoots       | Vera Anderson     | 2007-08-17 16:41:00
 Armada           | Snowboard      | Vera Anderson     | 2007-09-16 16:41:00
 Salomon          | Ski            | Allan Parker      | 2007-08-18 15:14:00
 Rossignol        | Snowboots      | Allan Parker      | 2007-08-25 14:14:00
 Nordica          | Snowboots      | Reginald Jackson  | 2007-05-20 12:47:00
(5 rows)

CREATE VIEW
 country  | num_people
------------+------------
 Turkey   |          4
 UAE      |          4
 USA      |          4
 Canada   |          4
 France   |          3
 China    |          3
 India    |          2
(7 rows)

   ptype         | numofbadproduct
-----------------+-----------------
 SkiBoots        |              13
 Snowboard       |              21
 Snowboots       |              17
 Poles           |              13
 Ski             |              14
(5 rows)

   brand      | amt
---------------+------
 Rossignol     | 1127
 Salomon       | 1219
 Armada        | 1931
(3 rows)

CREATE VIEW
CREATE VIEW
```

```
ptype          | majorage
----------------+------------
 Poles         |    2009
 SkiBoots     |    2008
 Ski              |    2012
 Snowboard |    2008
 Snowboots  |     2011
(5 rows)
```

6. Updates and Deletes

1)

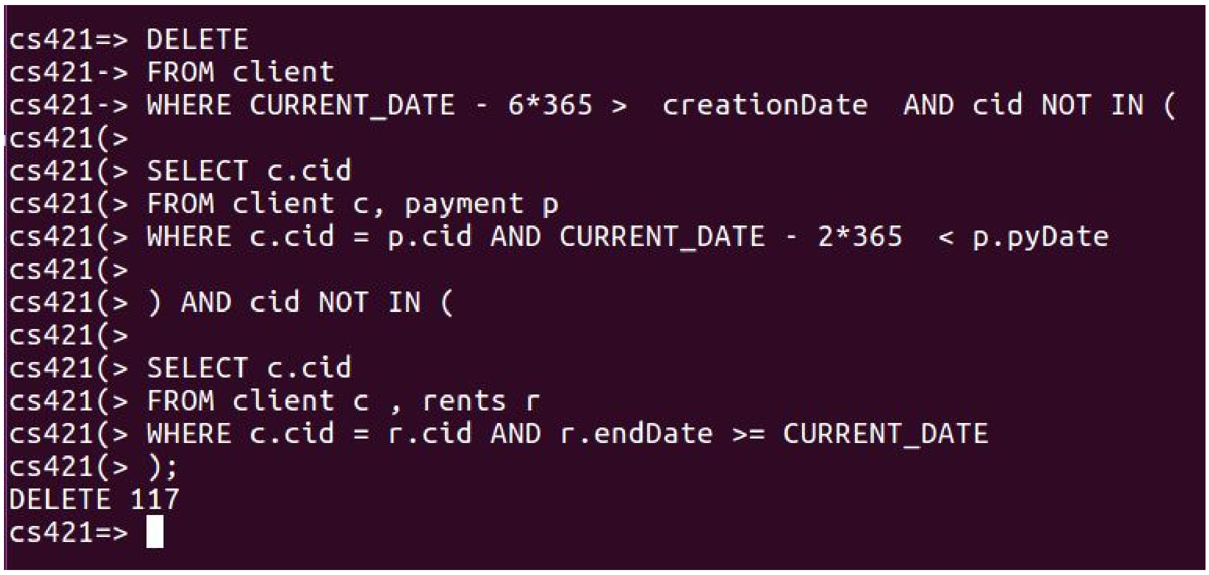
- Description :
  Deletes all client accounts that were created than 6 years ago or more, and that have been inactive (no payment) for 2 years, check that they are not currently renting equipment

- Tables involved : Client, Payment, Rents (See CREATE TABLE statements)
  Also affects : Rates by cascading

- Statements :

```
DELETE
FROM client
WHERE CURRENT_DATE - 6*365 >  creationDate  AND cid NOT IN (

SELECT c.cid
FROM client c, payment p
WHERE c.cid = p.cid AND CURRENT_DATE - 2*365  < p.pyDate

) AND cid NOT IN (

SELECT c.cid
FROM client c , rents r
WHERE c.cid = r.cid AND r.endDate >= CURRENT_DATE
);
```

- Execution:



```
cs421=> DELETE
cs421-> FROM client
cs421-> WHERE CURRENT_DATE - 6*365 >  creationDate  AND cid NOT IN (
cs421(>
cs421(> SELECT c.cid
cs421(> FROM client c, payment p
cs421(> WHERE c.cid = p.cid AND CURRENT_DATE - 2*365  < p.pyDate
cs421(>
cs421(> ) AND cid NOT IN (
cs421(>
cs421(> SELECT c.cid
cs421(> FROM client c , rents r
cs421(> WHERE c.cid = r.cid AND r.endDate >= CURRENT_DATE
cs421(> );
DELETE 117
cs421=>
```

2)

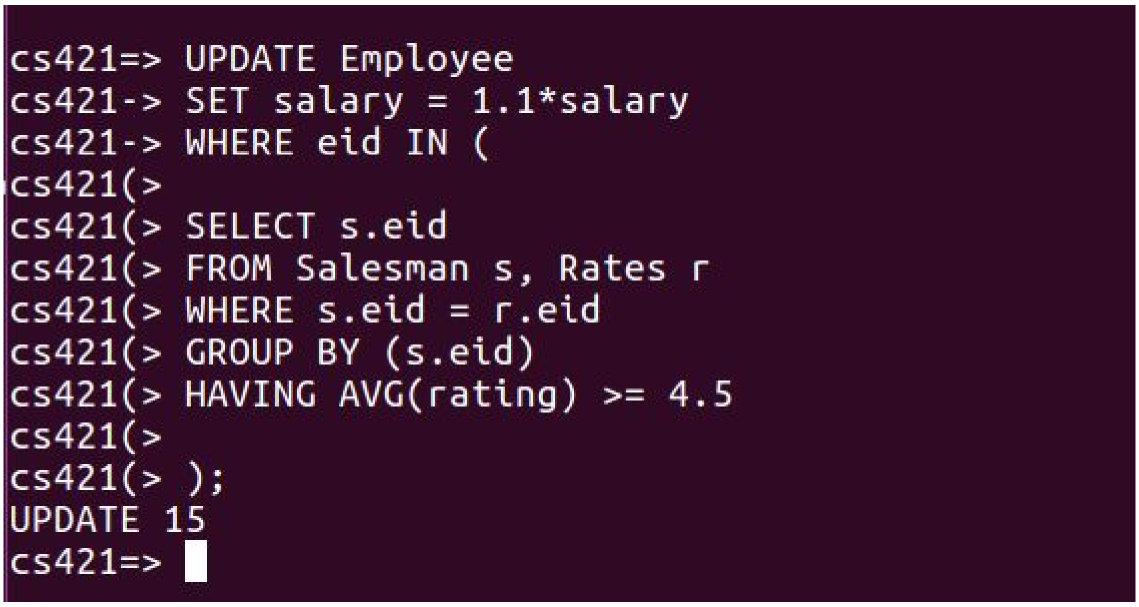
- Description:
  Give a 10% raise to all salesman with a rating over 4.5/5

- Tables involved :
  Salesman, Employee, Rates

- Statements :

```
UPDATE Employee
SET salary = 1.1*salary
WHERE eid IN (

       SELECT s.eid
       FROM Salesman s, Rates r
       WHERE s.eid = r.eid
       GROUP BY (s.eid)
       HAVING AVG(rating) >= 4.5

);
```

- Execution :



```
cs421=> UPDATE Employee
cs421-> SET salary = 1.1*salary
cs421-> WHERE eid IN (
cs421(>
cs421(> SELECT s.eid
cs421(> FROM Salesman s, Rates r
cs421(> WHERE s.eid = r.eid
cs421(> GROUP BY (s.eid)
cs421(> HAVING AVG(rating) >= 4.5
cs421(>
cs421(> );
UPDATE 15
cs421=>
```

3)

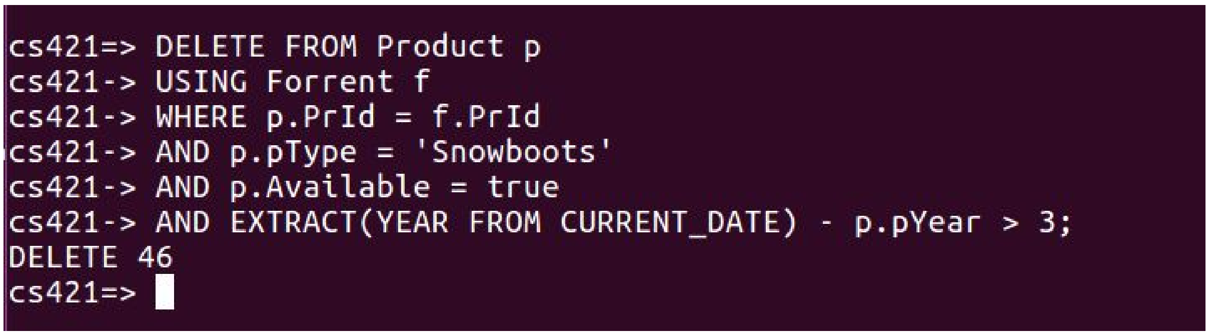
- Description :
  Delete all snowboots for rent that are 4 years old and more (check that they are not currently rented)

- Tables involved :
  Product, ForRent, Rents, PaysFor

- Statements :

```
DELETE FROM Product p
USING Forrent f
WHERE p.PrId = f.PrId
       AND p.pType = 'Snowboots'
       AND p.Available = true
       AND EXTRACT(YEAR FROM CURRENT_DATE) - p.pYear > 3;
```

- Execution :



```
cs421=> DELETE FROM Product p
cs421-> USING Forrent f
cs421-> WHERE p.PrId = f.PrId
cs421-> AND p.pType = 'Snowboots'
cs421-> AND p.Available = true
cs421-> AND EXTRACT(YEAR FROM CURRENT_DATE) - p.pYear > 3;
DELETE 46
cs421=>
```

4)

- Description :

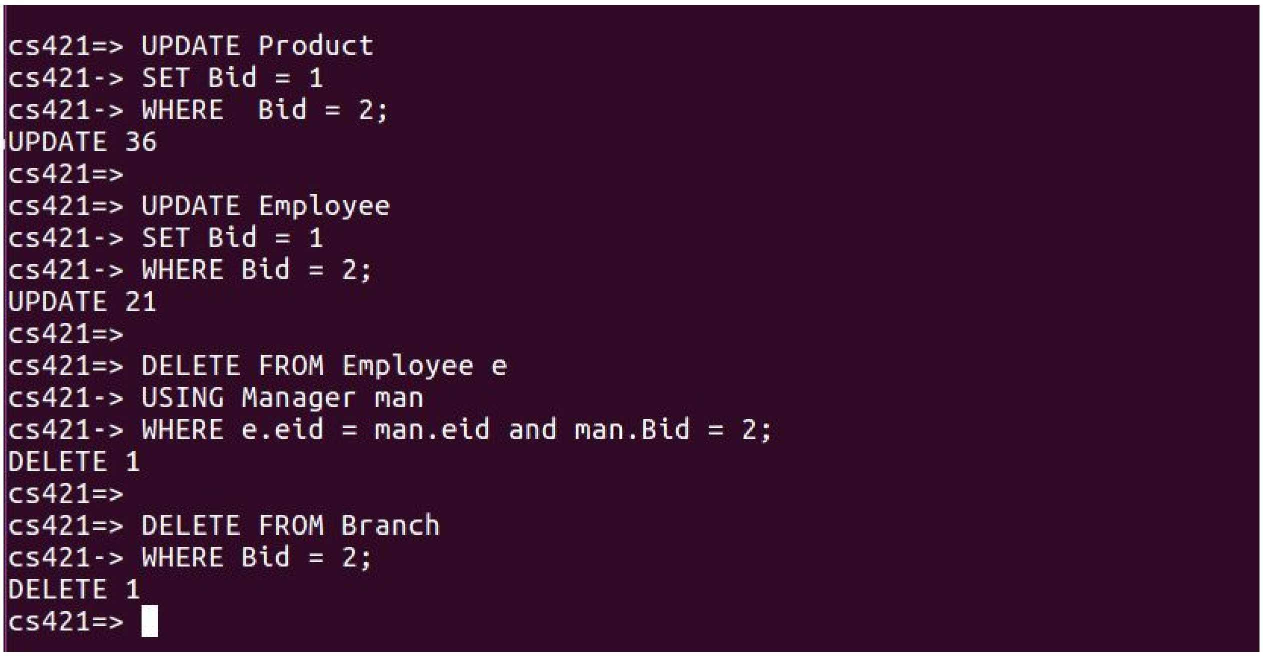
Merges branch with bid = 1 and bid = 2, make them one branch with bid = 1, delete the manager of the branch 2, delete the branch 2.

- Tables involved :
  Product, Employee, Manager, Branch

- Statements :

```
UPDATE Product
SET Bid = 1
WHERE  Bid = 2;

UPDATE Employee
SET Bid = 1
WHERE Bid = 2;

DELETE FROM Employee e
USING Manager man
WHERE e.eid = man.eid and man.Bid = 2;

DELETE FROM Branch
WHERE Bid = 2;
```

- Execution :



```
cs421=> UPDATE Product
cs421-> SET Bid = 1
cs421-> WHERE  Bid = 2;
UPDATE 36
cs421=>
cs421=> UPDATE Employee
cs421-> SET Bid = 1
cs421-> WHERE Bid = 2;
UPDATE 21
cs421=>
cs421=> DELETE FROM Employee e
cs421-> USING Manager man
cs421-> WHERE e.eid = man.eid and man.Bid = 2;
DELETE 1
cs421=>
cs421=> DELETE FROM Branch
cs421-> WHERE Bid = 2;
DELETE 1
cs421=>
```

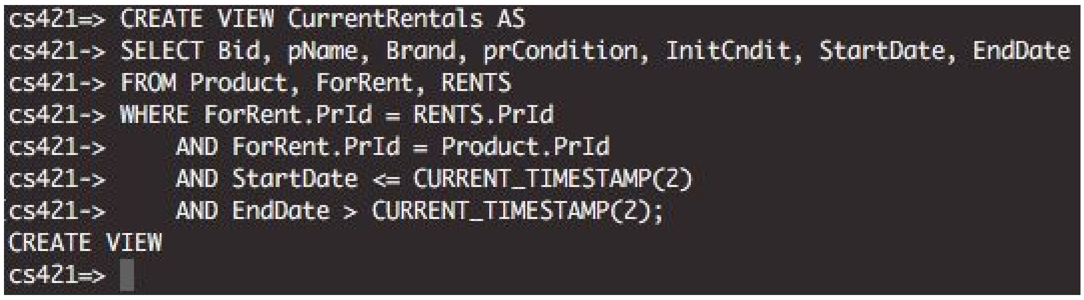
7. Views

View 1:

- Create SQL:

```
CREATE VIEW CurrentRentals AS
SELECT Bid, pName, Brand, prCondition, InitCndit, StartDate, EndDate
FROM Product, ForRent, RENTS
WHERE ForRent.PrId = RENTS.PrId
    AND ForRent.PrId = Product.PrId
    AND StartDate <= CURRENT_TIMESTAMP(2)
    AND EndDate > CURRENT_TIMESTAMP(2);
```

- Description:
  This view displays all currently "checked out" rental items, along with the branch they were rented from and the rental start and end timestamps.
- Output:

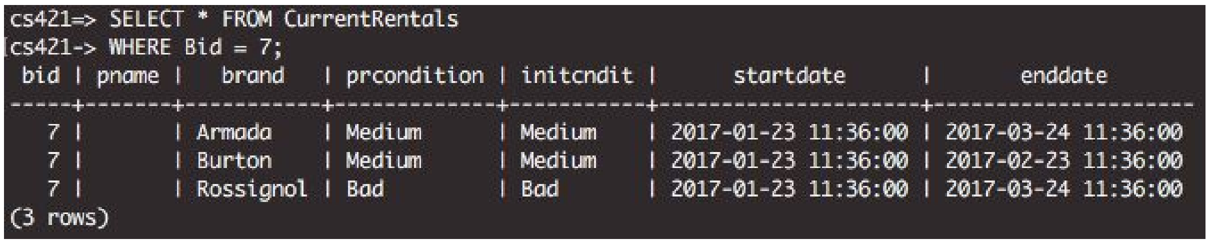


```
cs421=> CREATE VIEW CurrentRentals AS
cs421-> SELECT Bid, pName, Brand, prCondition, InitCndit, StartDate, EndDate
cs421-> FROM Product, ForRent, RENTS
cs421-> WHERE ForRent.PrId = RENTS.PrId
cs421->     AND ForRent.PrId = Product.PrId
cs421->     AND StartDate <= CURRENT_TIMESTAMP(2)
cs421->     AND EndDate > CURRENT_TIMESTAMP(2);
CREATE VIEW
cs421=>
```

- Use Case Query:

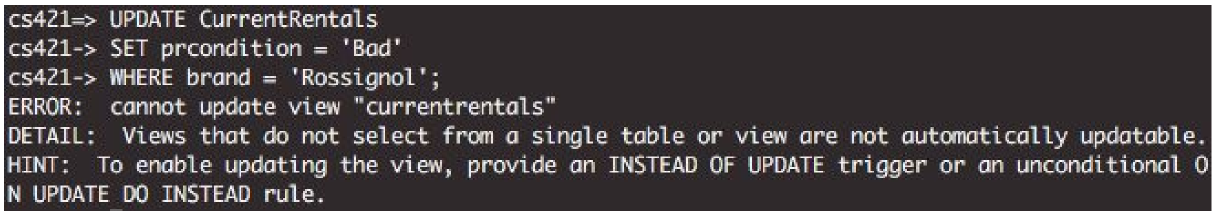
```
SELECT * FROM CurrentRentals
WHERE Bid = 7;
```

- Use Case Query Output:



```
cs421=> SELECT * FROM CurrentRentals
cs421-> WHERE Bid = 7;
 bid | pname |   brand   | prcondition | initcndit |      startdate      |       enddate
-----+-------+-----------+-------------+-----------+---------------------+---------------------
   7 |       | Armada    | Medium      | Medium    | 2017-01-23 11:36:00 | 2017-03-24 11:36:00
   7 |       | Burton    | Medium      | Medium    | 2017-01-23 11:36:00 | 2017-02-23 11:36:00
   7 |       | Rossignol | Bad         | Bad       | 2017-01-23 11:36:00 | 2017-03-24 11:36:00
(3 rows)
```

- Update Output:



```
cs421=> UPDATE CurrentRentals
cs421-> SET prcondition = 'Bad'
cs421-> WHERE brand = 'Rossignol';
ERROR:  cannot update view "currentrentals"
DETAIL:  Views that do not select from a single table or view are not automatically updatable.
HINT:  To enable updating the view, provide an INSTEAD OF UPDATE trigger or an unconditional O
N UPDATE DO INSTEAD rule.
```

- Update Explanation:
  This view is not updatable because it joins multiple tables. The following conditions must apply for a view to be updatable by default, quoting from the PostgreSQL Docs[1]:
  - *The view must have exactly one entry in its FROM list, which must be a table or another updatable view.*

[1] https://www.postgresql.org/docs/current/static/sql-createview.html#SQL-CREATEVIEW-UPDATABLE-VIEWS

- *The view definition must not contain WITH, DISTINCT, GROUP BY, HAVING, LIMIT, or OFFSET clauses at the top level.*
- *The view definition must not contain set operations (UNION, INTERSECT or EXCEPT) at the top level.*
- *The view's select list must not contain any aggregates, window functions or set-returning functions.*

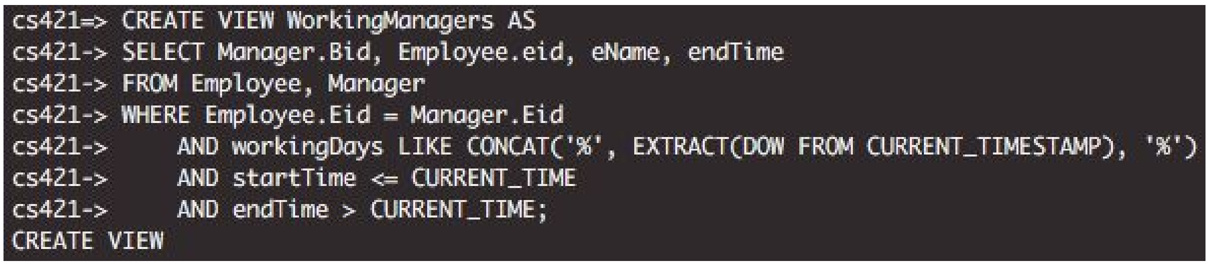
A view is also updatable if an (INSTEAD OF UPDATE) TRIGGER is created, or an (UPDATE) RULE is created with the CREATE statement in PostgreSQL. This is HINTed at when attempting to update this view.

View 2:

- Create SQL:

```
CREATE VIEW WorkingManagers AS
SELECT Manager.Bid, Employee.eid, eName, endTime
FROM Employee, Manager
WHERE Employee.Eid = Manager.Eid
  AND workingDays LIKE CONCAT('%', EXTRACT(DOW FROM CURRENT_TIMESTAMP), '%')
  AND startTime <= CURRENT_TIME
  AND endTime > CURRENT_TIME;
```

- Description:
  This view displays currently working managers and the open branches that they are managing.
- Output:

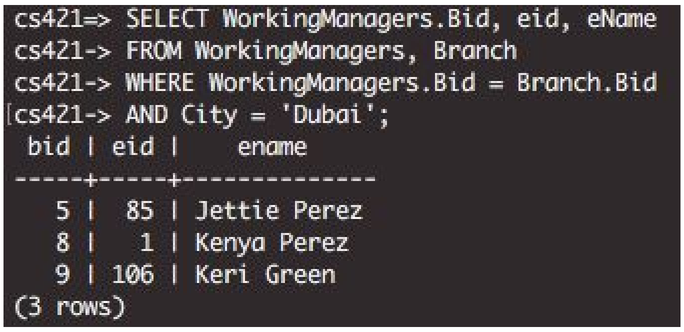


```
cs421=> CREATE VIEW WorkingManagers AS
cs421-> SELECT Manager.Bid, Employee.eid, eName, endTime
cs421-> FROM Employee, Manager
cs421-> WHERE Employee.Eid = Manager.Eid
cs421->     AND workingDays LIKE CONCAT('%', EXTRACT(DOW FROM CURRENT_TIMESTAMP), '%')
cs421->     AND startTime <= CURRENT_TIME
cs421->     AND endTime > CURRENT_TIME;
CREATE VIEW
```

- Use Case Query:

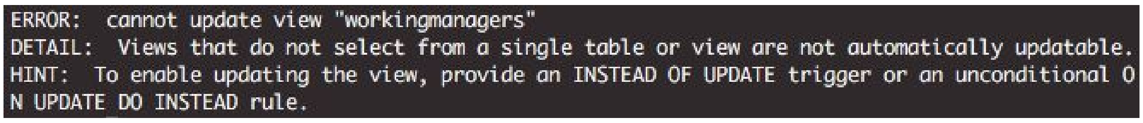
```
SELECT WorkingManagers.Bid, eid, eName
FROM WorkingManagers, Branch
WHERE WorkingManagers.Bid = Branch.Bid
AND City = 'Dubai';
```

- Use Case Query Output:



```
cs421=> SELECT WorkingManagers.Bid, eid, eName
cs421-> FROM WorkingManagers, Branch
cs421-> WHERE WorkingManagers.Bid = Branch.Bid
cs421-> AND City = 'Dubai';
 bid | eid |     ename
-----+-----+-------------
   5 |  85 | Jettie Perez
   8 |   1 | Kenya Perez
   9 | 106 | Keri Green
(3 rows)
```

- Update Output:



```
ERROR:  cannot update view "workingmanagers"
DETAIL:  Views that do not select from a single table or view are not automatically updatable.
HINT:  To enable updating the view, provide an INSTEAD OF UPDATE trigger or an unconditional O
N UPDATE DO INSTEAD rule.
```

- Update Explanation:
  This view is not updatable because it joins multiple tables. See “Update Explanation” for View 1 for further explanation.

8. Check Constraints

Some check constraints were already here before, but we added 2 new constraints to make it even more consistent.

- Time constraint for opening times and closing times :

```
CREATE TABLE Branch(
       Bid INTEGER PRIMARY KEY,
       StreetNumber INTEGER,
       Street VARCHAR(30),
       City VARCHAR(30),
       Country VARCHAR(20),
       OpeningTime TIME,
       ClosingTime TIME CHECK(closingTime > openingTime)
);
```

- Unsuccessful inserts :

```
INSERT INTO branch VALUES(1, 1234, 'St. Marc', 'Montreal', 'Canada', '18:00', '10:00');
```

- Output :

```
ERROR:  new row for relation "branch" violates check constraint "branch_check"
DETAIL:  Failing row contains (1, 1234, St. Marc, Montreal, Canada, 18:00:00, 10:00:00).

********** Error **********

ERROR: new row for relation "branch" violates check constraint "branch_check"
SQL state: 23514
Detail: Failing row contains (1, 1234, St. Marc, Montreal, Canada, 18:00:00, 10:00:00).
```

- Time constraint for rental periods :

```
CREATE TABLE RENTS(
       CONSTRAINT RentId   PRIMARY KEY(Cid,PyId,PrId),
       Cid    INTEGER      NOT NULL REFERENCES Client(Cid) ON DELETE CASCADE ON UPDATE
CASCADE,
       PyId   INTEGER      NOT NULL REFERENCES Payment(PyId) ON DELETE CASCADE ON UPDATE
CASCADE,
       PrId   INTEGER      NOT NULL REFERENCES ForRent(PrId) ON DELETE CASCADE ON UPDATE
CASCADE,

       InitCndit   VARCHAR(50)    NOT NULL,
       StartDate   TIMESTAMP   NOT NULL DEFAULT   CURRENT_TIMESTAMP(2),
       EndDate       TIMESTAMP   NOT NULL CHECK (endDate > startDate)

);
```

- <u>Statement:</u>

```
INSERT INTO rents VALUES(1,1,1, 'GOOD', '2016-01-01 08:00', '2015-01-01 08:00');
```

- <u>Output:</u>

```
ERROR:  new row for relation "rents" violates check constraint "rents_check"
DETAIL:  Failing row contains (1, 1, 1, GOOD, 2016-01-01 08:00:00, 2015-01-01 08:00:00).

********** Error **********

ERROR: new row for relation "rents" violates check constraint "rents_check"
SQL state: 23514
Detail: Failing row contains (1, 1, 1, GOOD, 2016-01-01 08:00:00, 2015-01-01 08:00:00).
```

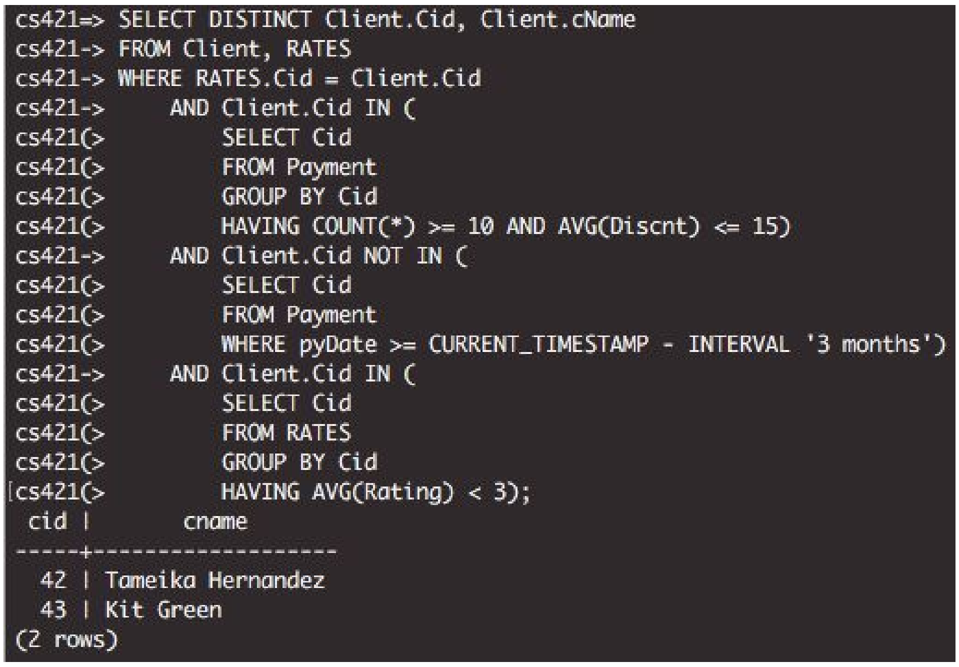
9. Creativity Points

- **Data Generation**:
  Data was inserted in the database using an Automated program that creates INSERT statements in Scala programming language. The program generated 10 branches, 200 employees, 300 clients, 400 products and 100 payments. The program takes care of consistency in the data (for example the due dates of rentals, the total amount of a payment) and generates data randomly based on initial data.

- **Analytical Query:**

```
SELECT DISTINCT Client.Cid, Client.cName
FROM Client, RATES
WHERE RATES.Cid = Client.Cid
    AND Client.Cid IN (
        SELECT Cid
        FROM Payment
        GROUP BY Cid
        HAVING COUNT(*) >= 10 AND AVG(Discnt) <= 15)
    AND Client.Cid NOT IN (
        SELECT Cid
        FROM Payment
        WHERE pyDate >= CURRENT_TIMESTAMP - INTERVAL '3 months')
    AND Client.Cid IN (
        SELECT Cid
        FROM RATES
        GROUP BY Cid
        HAVING AVG(Rating) < 3);
```

- Execution:



```
cs421=> SELECT DISTINCT Client.Cid, Client.cName
cs421-> FROM Client, RATES
cs421-> WHERE RATES.Cid = Client.Cid
cs421->     AND Client.Cid IN (
cs421(>         SELECT Cid
cs421(>         FROM Payment
cs421(>         GROUP BY Cid
cs421(>         HAVING COUNT(*) >= 10 AND AVG(Discnt) <= 15)
cs421->     AND Client.Cid NOT IN (
cs421(>         SELECT Cid
cs421(>         FROM Payment
cs421(>         WHERE pyDate >= CURRENT_TIMESTAMP - INTERVAL '3 months')
cs421->     AND Client.Cid IN (
cs421(>         SELECT Cid
cs421(>         FROM RATES
cs421(>         GROUP BY Cid
cs421(>         HAVING AVG(Rating) < 3);
 cid |      cname
-----+------------------
  42 | Tameika Hernandez
  43 | Kit Green
(2 rows)
```

- Description:

  The above query addresses the business requirement of finding previously loyal customers who went unnoticed or underserviced by salespeople. It displays high-volume clients who have given a lower overall rating to the salespeople who served them, have received few discounts despite their patronage and dissatisfaction, and who presumably as a result have recently stopped shopping at the store.