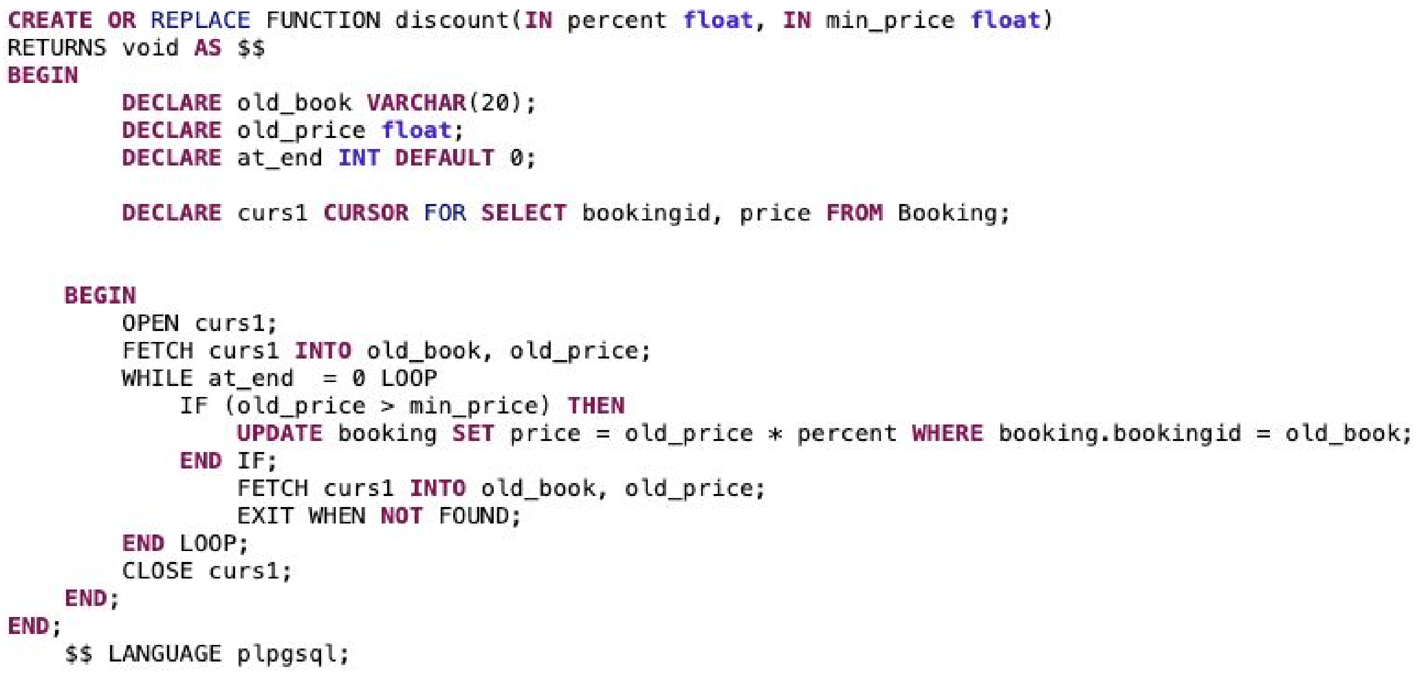
COMP 421 Database Systems   Project Deliverable#03   Group#55

Members: Jonathan Ng, Melody Song, Sunny Kim, Van Vai Vivien Lee
Database: PostgreSQL

Q1. Stored Procedure

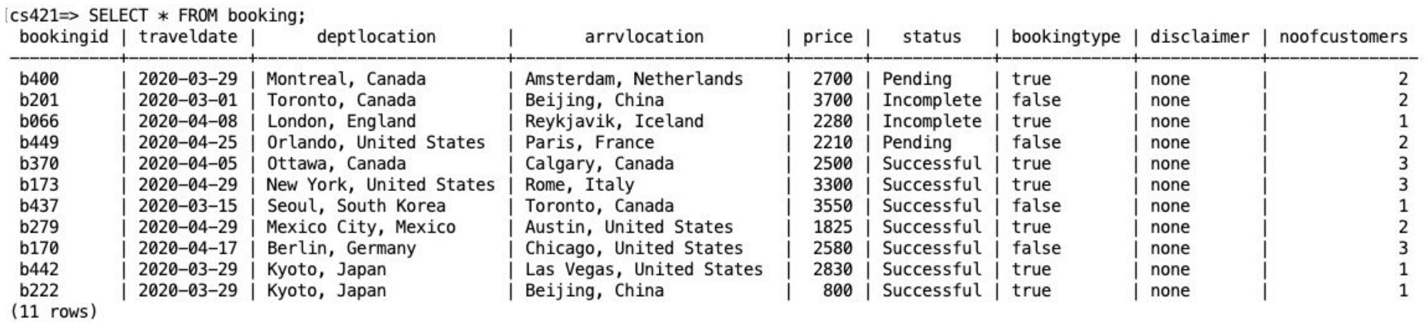
This function applies a discount to bookings that exceed a certain price . There are two float input arguments-- 1) discount percentage; 2) minimum price to qualify for the booking.

Stored Procedure



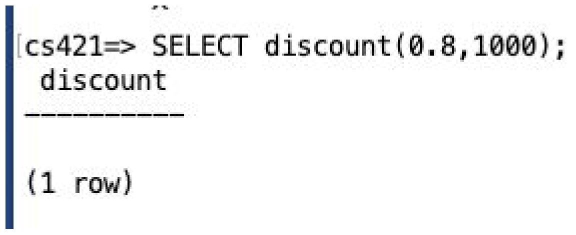
```
CREATE OR REPLACE FUNCTION discount(IN percent float, IN min_price float)
RETURNS void AS $$
BEGIN
        DECLARE old_book VARCHAR(20);
        DECLARE old_price float;
        DECLARE at_end INT DEFAULT 0;

        DECLARE curs1 CURSOR FOR SELECT bookingid, price FROM Booking;


    BEGIN
        OPEN curs1;
        FETCH curs1 INTO old_book, old_price;
        WHILE at_end  = 0 LOOP
            IF (old_price > min_price) THEN
                UPDATE booking SET price = old_price * percent WHERE booking.bookingid = old_book;
            END IF;
                FETCH curs1 INTO old_book, old_price;
                EXIT WHEN NOT FOUND;
        END LOOP;
        CLOSE curs1;
    END;
END;
    $$ LANGUAGE plpgsql;
```

Before the function discount is called, the records of booking are as below:



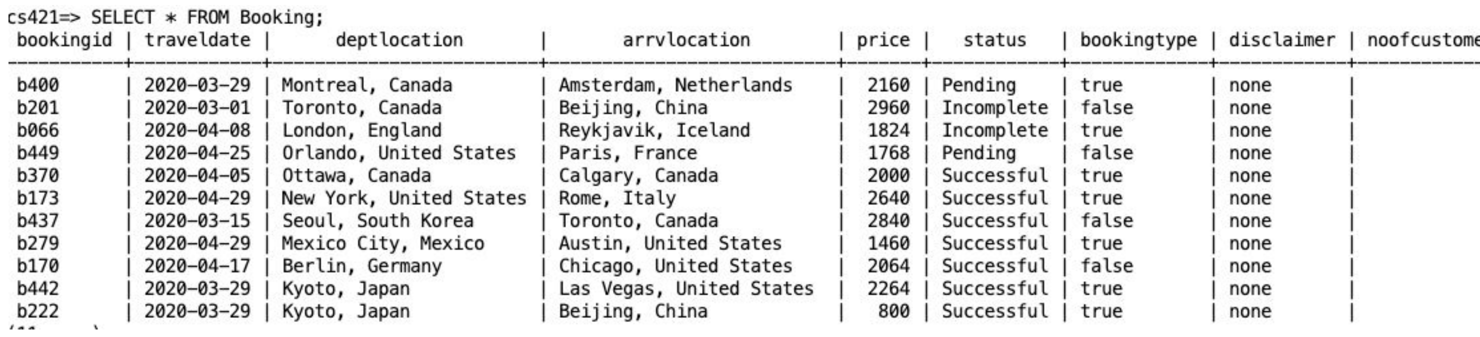
```
cs421=> SELECT * FROM booking;
 bookingid | traveldate |      deptlocation       |        arrvlocation        | price |   status   | bookingtype | disclaimer | noofcustomers
-----------+------------+-------------------------+----------------------------+-------+------------+-------------+------------+---------------
 b400      | 2020-03-29 | Montreal, Canada        | Amsterdam, Netherlands     |  2700 | Pending    | true        | none       |             2
 b201      | 2020-03-01 | Toronto, Canada         | Beijing, China             |  3700 | Incomplete | false       | none       |             2
 b066      | 2020-04-08 | London, England         | Reykjavik, Iceland         |  2280 | Incomplete | true        | none       |             1
 b449      | 2020-04-25 | Orlando, United States  | Paris, France              |  2210 | Pending    | false       | none       |             2
 b370      | 2020-04-05 | Ottawa, Canada          | Calgary, Canada            |  2500 | Successful | true        | none       |             3
 b173      | 2020-04-29 | New York, United States | Rome, Italy                |  3300 | Successful | true        | none       |             3
 b437      | 2020-03-15 | Seoul, South Korea      | Toronto, Canada            |  3550 | Successful | false       | none       |             1
 b279      | 2020-04-29 | Mexico City, Mexico     | Austin, United States      |  1825 | Successful | true        | none       |             2
 b170      | 2020-04-17 | Berlin, Germany         | Chicago, United States     |  2580 | Successful | false       | none       |             3
 b442      | 2020-03-29 | Kyoto, Japan            | Las Vegas, United States   |  2830 | Successful | true        | none       |             1
 b222      | 2020-03-29 | Kyoto, Japan            | Beijing, China             |   800 | Successful | true        | none       |             1
(11 rows)
```

Now we test this function passing 0.8 (80% of original price) and 1000 as arguments:



```
cs421=> SELECT discount(0.8,1000);
 discount
----------

(1 row)
```

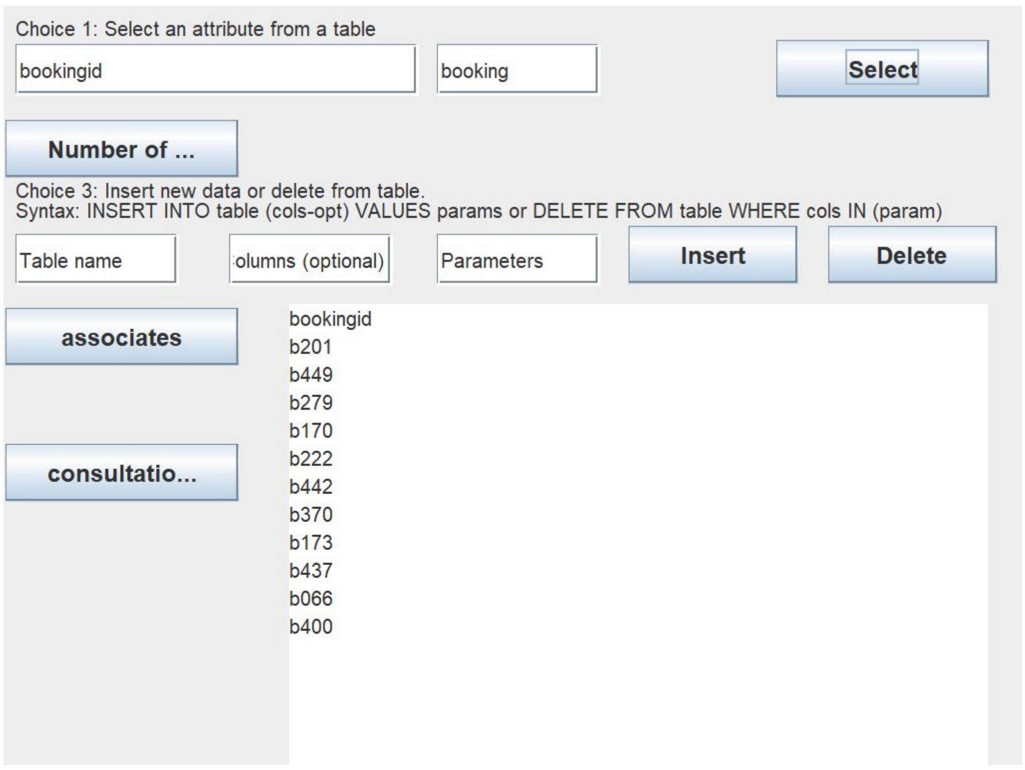
The output records show that bookings with original price > 1000 got a 20% discount



```
cs421=> SELECT * FROM Booking;
 bookingid | traveldate |      deptlocation        |       arrvlocation        | price |   status    | bookingtype | disclaimer | noofcustome
-----------+------------+--------------------------+---------------------------+-------+-------------+-------------+------------+------------
 b400      | 2020-03-29 | Montreal, Canada         | Amsterdam, Netherlands    |  2160 | Pending     | true        | none       |
 b201      | 2020-03-01 | Toronto, Canada          | Beijing, China            |  2960 | Incomplete  | false       | none       |
 b066      | 2020-04-08 | London, England          | Reykjavik, Iceland        |  1824 | Incomplete  | true        | none       |
 b449      | 2020-04-25 | Orlando, United States   | Paris, France             |  1768 | Pending     | false       | none       |
 b370      | 2020-04-05 | Ottawa, Canada           | Calgary, Canada           |  2000 | Successful  | true        | none       |
 b173      | 2020-04-29 | New York, United States  | Rome, Italy               |  2640 | Successful  | true        | none       |
 b437      | 2020-03-15 | Seoul, South Korea       | Toronto, Canada           |  2840 | Successful  | false       | none       |
 b279      | 2020-04-29 | Mexico City, Mexico      | Austin, United States     |  1460 | Successful  | true        | none       |
 b170      | 2020-04-17 | Berlin, Germany          | Chicago, United States    |  2064 | Successful  | false       | none       |
 b442      | 2020-03-29 | Kyoto, Japan             | Las Vegas, United States  |  2264 | Successful  | true        | none       |
 b222      | 2020-03-29 | Kyoto, Japan             | Beijing, China            |   800 | Successful  | true        | none       |
```

2. Java Application

1) Select a single attribute from a table


Choice 1: Select an attribute from a table
bookingid
booking
Select
Number of ...
Choice 3: Insert new data or delete from table.
Syntax: INSERT INTO table (cols-opt) VALUES params or DELETE FROM table WHERE cols IN (param)
Table name
:olumns (optional)
Parameters
Insert
Delete
associates
consultatio...

```
bookingid
b201
b449
b279
b170
b222
b442
b370
b173
b437
b066
b400
```

By inputting the table name and parameters to the fields, we can acquire a single column from any table

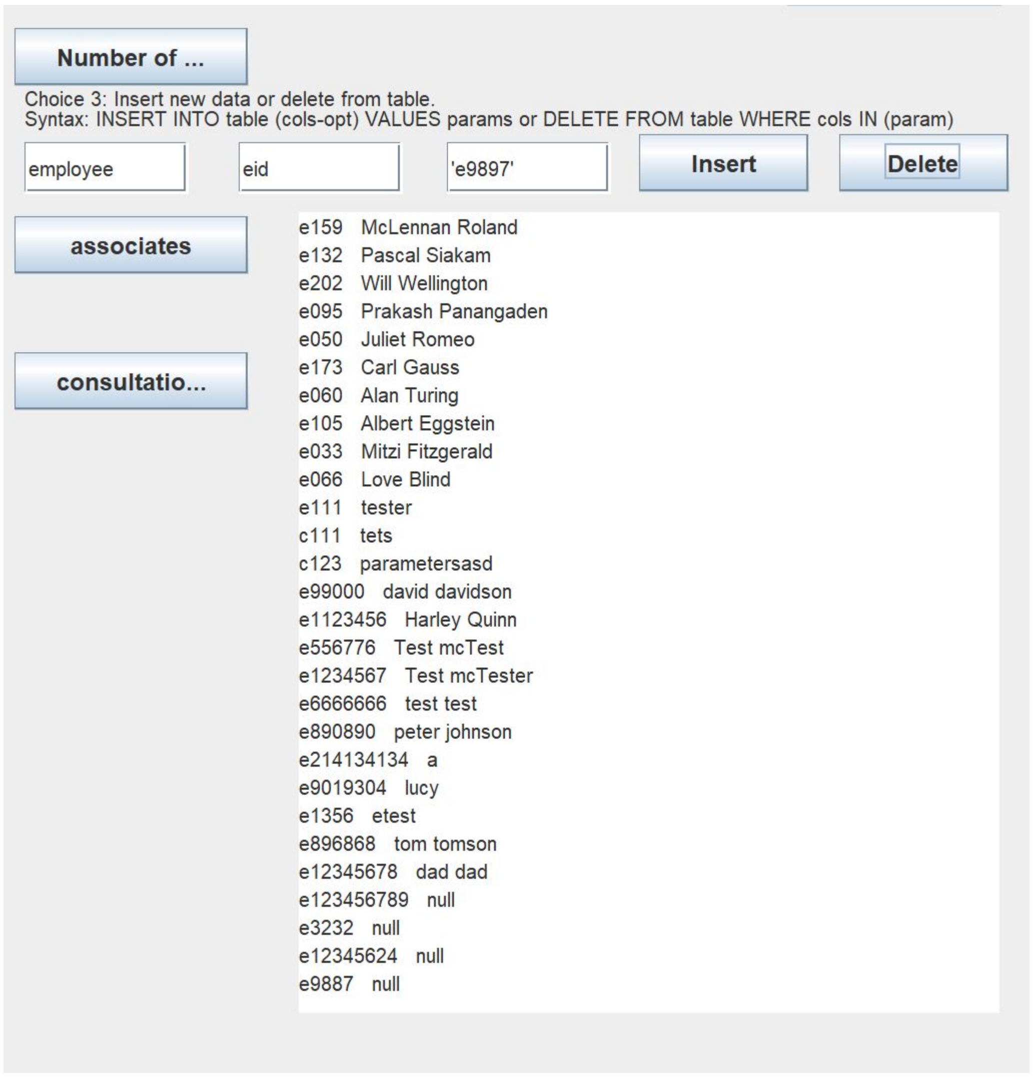
2) We find customers without insurance by doing two parameters except the other

SELECT customer.custid FROM customer
EXCEPT
SELECT associates.custid FROM associates;

3) Inserting & Deleting a new record to & from a table

Insert: Using employee as an example, inserting a new employee record

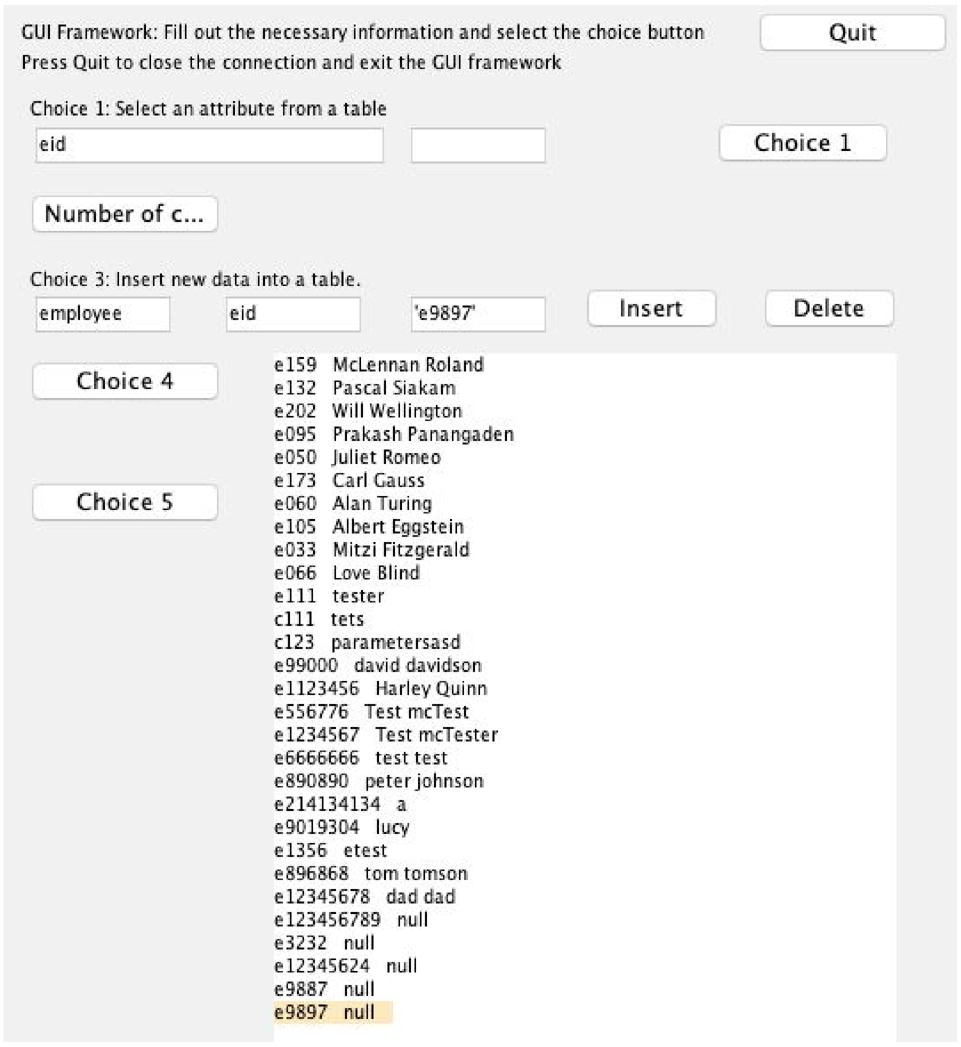
- Before insertion (only displayed full records, i.e. records don't have null values)



Choice 3: Insert new data or delete from table.
Syntax: INSERT INTO table (cols-opt) VALUES params or DELETE FROM table WHERE cols IN (param)

```
e159   McLennan Roland
e132   Pascal Siakam
e202   Will Wellington
e095   Prakash Panangaden
e050   Juliet Romeo
e173   Carl Gauss
e060   Alan Turing
e105   Albert Eggstein
e033   Mitzi Fitzgerald
e066   Love Blind
e111   tester
c111   tets
c123   parametersasd
e99000   david davidson
e1123456   Harley Quinn
e556776   Test mcTest
e1234567   Test mcTester
e6666666   test test
e890890   peter johnson
e214134134   a
e9019304   lucy
e1356   etest
e896868   tom tomson
e12345678   dad dad
e123456789   null
e3232   null
e12345624   null
e9887   null
```

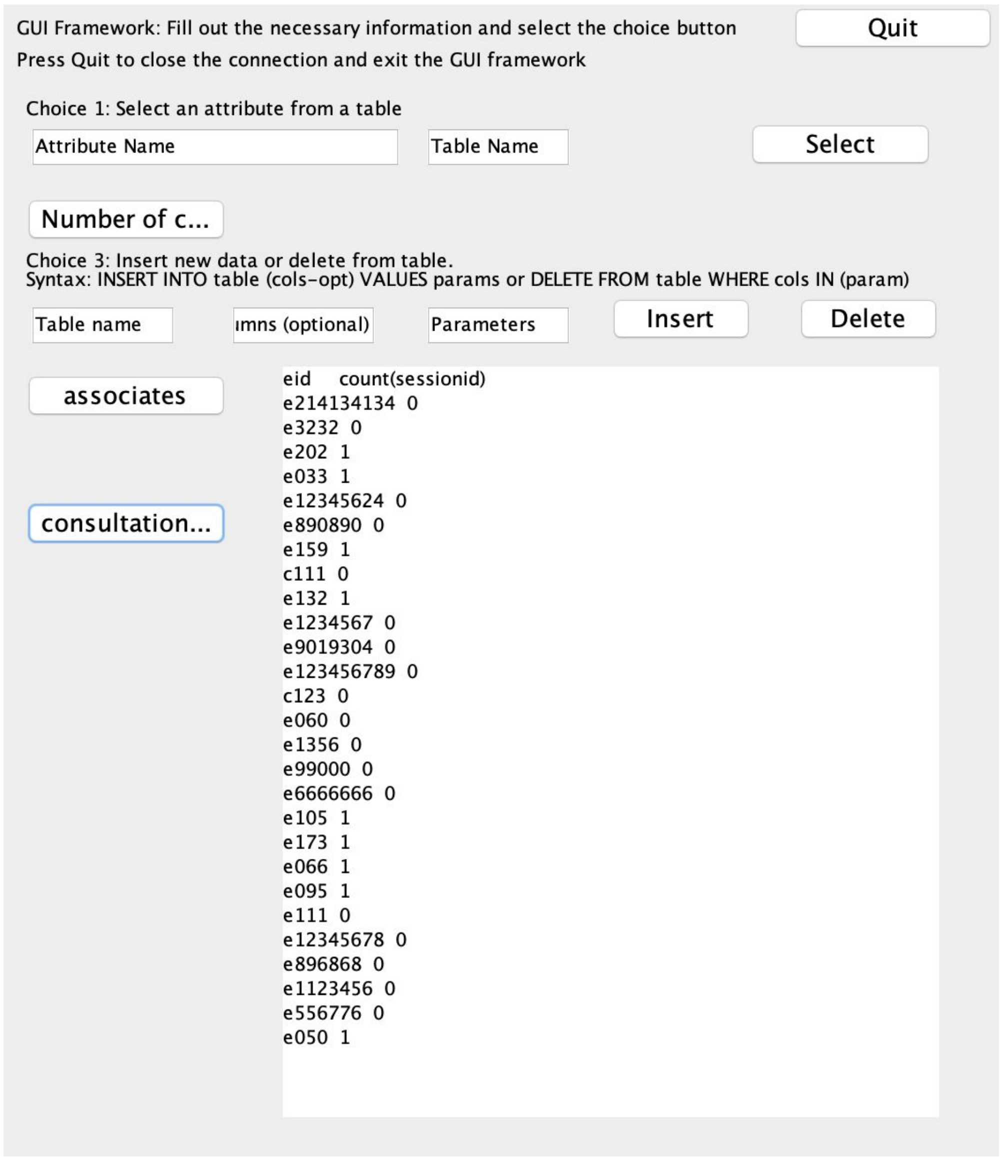
- After insertion: we inserted eid = 'e9897'


GUI Framework: Fill out the necessary information and select the choice button
Press Quit to close the connection and exit the GUI framework
Quit
Choice 1: Select an attribute from a table
eid
Choice 1
Number of c...
Choice 3: Insert new data into a table.
employee
eid
'e9897'
Insert
Delete
Choice 4
Choice 5

```
e159   McLennan Roland
e132   Pascal Siakam
e202   Will Wellington
e095   Prakash Panangaden
e050   Juliet Romeo
e173   Carl Gauss
e060   Alan Turing
e105   Albert Eggstein
e033   Mitzi Fitzgerald
e066   Love Blind
e111   tester
c111   tets
c123   parametersasd
e99000   david davidson
e1123456   Harley Quinn
e556776   Test mcTest
e1234567   Test mcTester
e6666666   test test
e890890   peter johnson
e214134134   a
e9019304   lucy
e1356   etest
e896868   tom tomson
e12345678   dad dad
e123456789   null
e3232   null
e12345624   null
e9887   null
e9897   null
```

5)Counting the # of consultations each employee has

```
SELECT eid, COUNT(sessionid) FROM
(SELECT employee.eid, consults.sessionid FROM employee
LEFT JOIN consults on
employee.eid = consults.eid) as A
GROUP BY A.eid;
```


GUI Framework: Fill out the necessary information and select the choice button
Press Quit to close the connection and exit the GUI framework
Quit
Choice 1: Select an attribute from a table
Attribute Name
Table Name
Select
Number of c...
Choice 3: Insert new data or delete from table.
Syntax: INSERT INTO table (cols-opt) VALUES params or DELETE FROM table WHERE cols IN (param)
Table name
ımns (optional)
Parameters
Insert
Delete
associates
consultation...

```
eid    count(sessionid)
e214134134  0
e3232  0
e202  1
e033  1
e12345624  0
e890890  0
e159  1
c111  0
e132  1
e1234567  0
e9019304  0
e123456789  0
c123  0
e060  0
e1356  0
e99000  0
e6666666  0
e105  1
e173  1
e066  1
e095  1
e111  0
e12345678  0
e896868  0
e1123456  0
e556776  0
e050  1
```

3. Creating indices

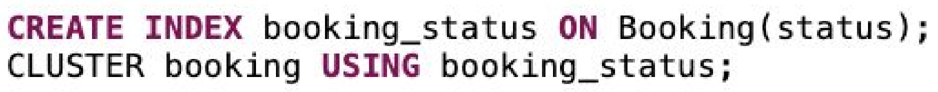
**Note: Unlike DB2, postgreSQL does not have the option to create a clustered index in the index definition.*

a. Index on attribute status of relation booking (clustered)

- The primary key of booking is bookingid

This index is useful because a travel agency will oftentimes have to track the status of a booking and see if further actions need to be executed. To recall, there are three possible values for booking status-- “pending”, “incomplete” and “successful”. By using an index on this attribute we can quickly select pages with matching records.

For example, if we do SELECT * FROM Booking WHERE status = “successful”. Instead of having to scan all the data pages, we can read one leaf page (possibly more if it spreads over more than one leaf node) and corresponding data pages. Since we have a clustered index, assuming we have 30% matching records, we will only have to read 30% of the data pages to retrieve all th records. The total IO cost from reading will go down from 100% data pages to 30%data pages plus one leaf page.



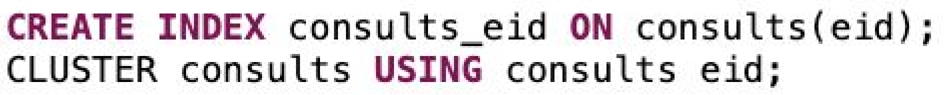
```
CREATE INDEX booking_status ON Booking(status);
CLUSTER booking USING booking_status;
```

b. Index on attribute eid of relation consults (unclustered)

- The primary key of consults is (customer id, eid)

This index will be useful if the travel agency wants to see how many consultation sessions a particular employee has conducted have conducted.

For example, if we do SELECT COUNT(sessionid) FROM consults WHERE eid = @xyz. If we have the index on eid (employee id), Instead of having to scan all the data pages in the relation, we can read one leaf page and corresponding data pages only. We will use a clustered index so all records belonging to the same employee are next to each other. This can bring our IO cost down from reading all data pages to possibly 1 leaf page and say only 1 data page (assuming all the matching records of this employee can fit into one data age).



```
CREATE INDEX consults_eid ON consults(eid);
CLUSTER consults USING consults eid;
```

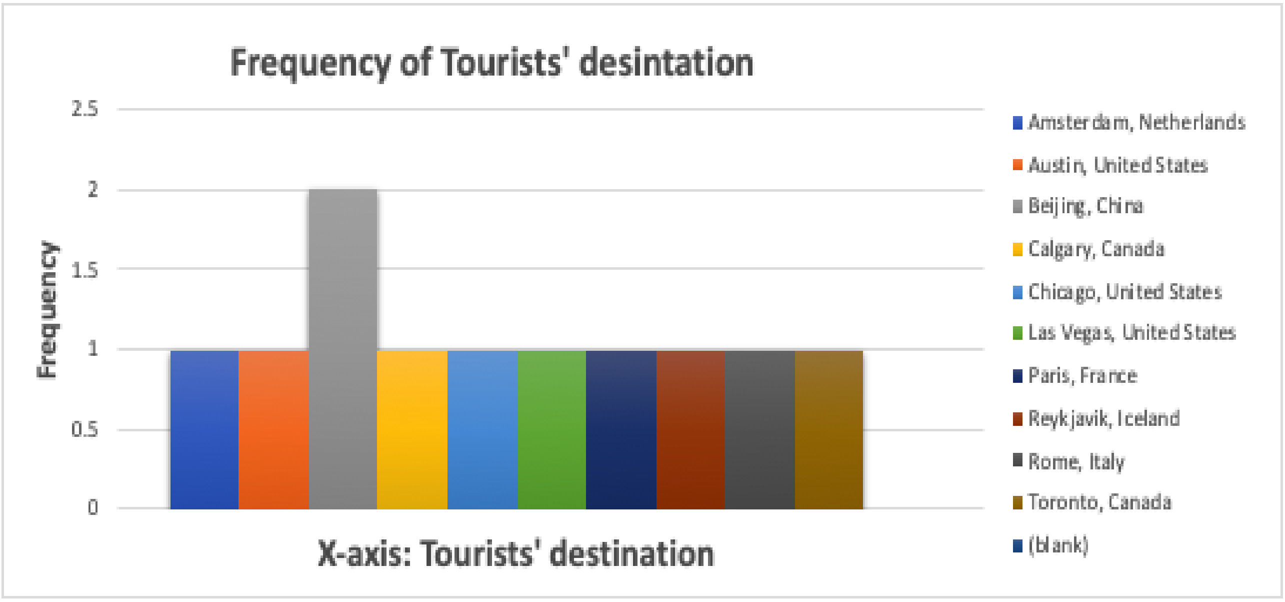
4. Data visualization

a. Frequency of Tourists’ destination

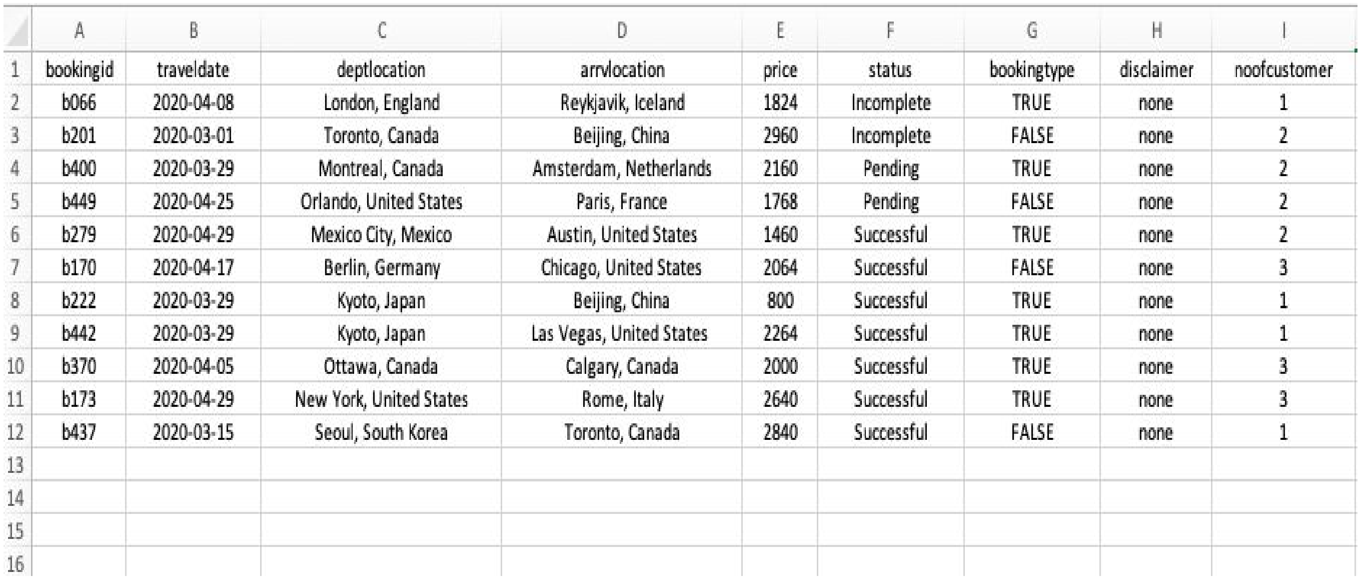
1) SQL Query to generate data (Relation booking)

```
cs421=> \COPY (SELECT* FROM booking) TO booking.csv WITH CSV
```

2) Excel pivot chart image



3) Excel file



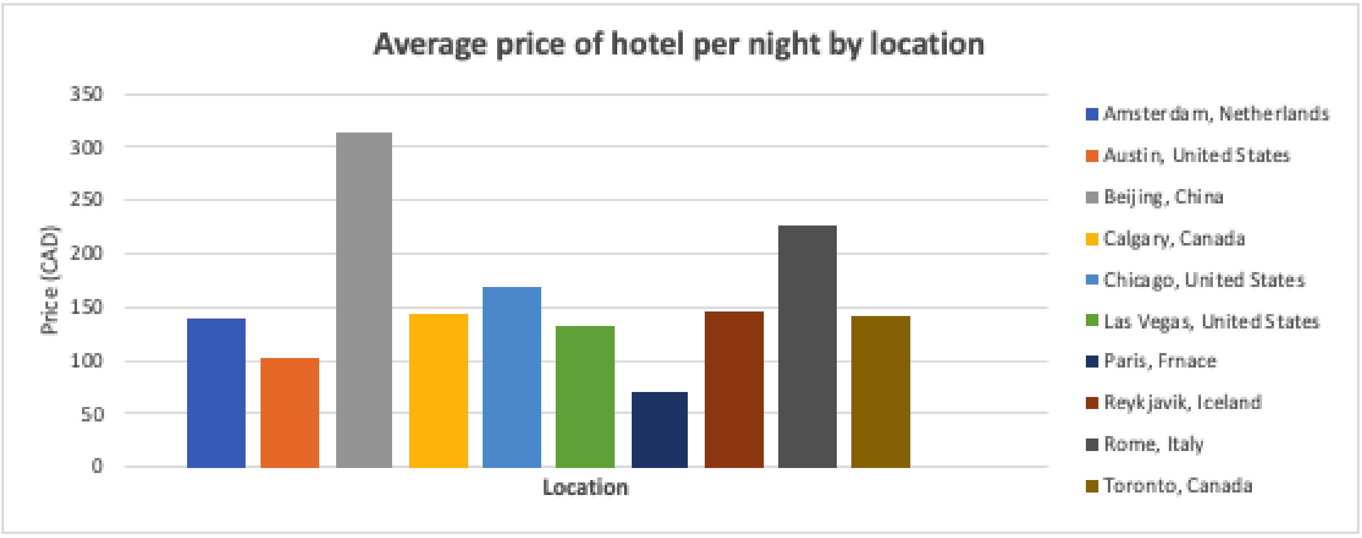
	A	B	C	D	E	F	G	H	I
1	bookingid	traveldate	deptlocation	arrvlocation	price	status	bookingtype	disclaimer	noofcustomer
2	b066	2020-04-08	London, England	Reykjavik, Iceland	1824	Incomplete	TRUE	none	1
3	b201	2020-03-01	Toronto, Canada	Beijing, China	2960	Incomplete	FALSE	none	2
4	b400	2020-03-29	Montreal, Canada	Amsterdam, Netherlands	2160	Pending	TRUE	none	2
5	b449	2020-04-25	Orlando, United States	Paris, France	1768	Pending	FALSE	none	2
6	b279	2020-04-29	Mexico City, Mexico	Austin, United States	1460	Successful	TRUE	none	2
7	b170	2020-04-17	Berlin, Germany	Chicago, United States	2064	Successful	FALSE	none	3
8	b222	2020-03-29	Kyoto, Japan	Beijing, China	800	Successful	TRUE	none	1
9	b442	2020-03-29	Kyoto, Japan	Las Vegas, United States	2264	Successful	TRUE	none	1
10	b370	2020-04-05	Ottawa, Canada	Calgary, Canada	2000	Successful	TRUE	none	3
11	b173	2020-04-29	New York, United States	Rome, Italy	2640	Successful	TRUE	none	3
12	b437	2020-03-15	Seoul, South Korea	Toronto, Canada	2840	Successful	FALSE	none	1
13									
14									
15									
16									

b. Average price of hotel per night by location

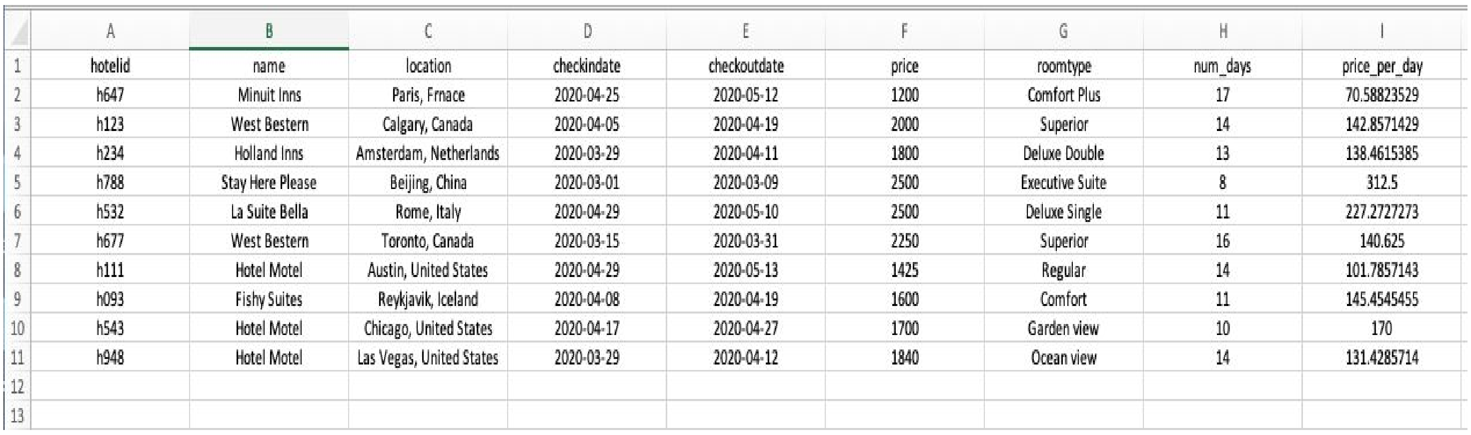
1) SQL Query to generate data (relation Hotel)

```
cs421=> \COPY (SELECT* FROM hotel) TO hotel.csv WITH CSV
```

2) Excel pivot chart image



3) Excel file



	A	B	C	D	E	F	G	H	I
1	hotelid	name	location	checkindate	checkoutdate	price	roomtype	num_days	price_per_day
2	h647	Minuit Inns	Paris, Frnace	2020-04-25	2020-05-12	1200	Comfort Plus	17	70.58823529
3	h123	West Bestern	Calgary, Canada	2020-04-05	2020-04-19	2000	Superior	14	142.8571429
4	h234	Holland Inns	Amsterdam, Netherlands	2020-03-29	2020-04-11	1800	Deluxe Double	13	138.4615385
5	h788	Stay Here Please	Beijing, China	2020-03-01	2020-03-09	2500	Executive Suite	8	312.5
6	h532	La Suite Bella	Rome, Italy	2020-04-29	2020-05-10	2500	Deluxe Single	11	227.2727273
7	h677	West Bestern	Toronto, Canada	2020-03-15	2020-03-31	2250	Superior	16	140.625
8	h111	Hotel Motel	Austin, United States	2020-04-29	2020-05-13	1425	Regular	14	101.7857143
9	h093	Fishy Suites	Reykjavik, Iceland	2020-04-08	2020-04-19	1600	Comfort	11	145.4545455
10	h543	Hotel Motel	Chicago, United States	2020-04-17	2020-04-27	1700	Garden view	10	170
11	h948	Hotel Motel	Las Vegas, United States	2020-03-29	2020-04-12	1840	Ocean view	14	131.4285714
12									
13									

*The last two columns- num_days and price_per_day are additional manipulations done in excel

5. Creativity points

a. Additional stored procedure

This function takes in nothing as an input but computes the sum (aggregate) of prices of each booking in the Booking relation. This is important to help the business learn how much sales they are generating in total.

```
CREATE OR REPLACE FUNCTION sum_booking_price() RETURNS float AS $total$
DECLARE total float;
BEGIN
    SELECT SUM(Booking.price) into total FROM Booking;
    RETURN total;
END;
$total$ LANGUAGE plpgsql;
```