Aggregation

1.Count the number of facilities

Query: select count(*) from cd.facilities;

SQLOUTPUT:

```
exercises=# select count(*) from cd.facilities;
 count
-------
    11
(1 row)
```

2.Count the number of expensive facilities

Query: select count(*) from cd.facilities where guestcost >= 10;

SQLOUTPUT:

```
exercises=# select count(*) from cd.facilities where guestcost >= 10;
 count
-------
     8
(1 row)
```

3.Count the number of recommendations each member makes.

Query: select recommendedby, count(*) from cd.members

where recommendedby

is not null group by recommendedby

order by recommendedby;

SQLOUTPUT:

```
exercises=#  select recommendedby, count(*) from cd.members
exercises-# where recommendedby
exercises-# is not null group by recommendedby
exercises-# order by recommendedby;
 recommendedby | count
---------------+-------
             1 |     5
             2 |     3
             3 |     1
             4 |     2
             5 |     1
             6 |     1
             9 |     2
            11 |     1
            13 |     2
            15 |     1
            16 |     1
            20 |     1
            30 |     1
(13 rows)
```

4.List the total slots booked per facility

Query: select facid, sum(slots) as "Total Slots"

from cd.bookings

group by facid

order by facid;

SQLOUTPUT:

```
exercises=# select facid, sum(slots) as "Total Slots"
exercises-# from cd.bookings
exercises-# group by facid
exercises-#          order by facid;
 facid | Total Slots
-------+-------------
     0 |        1320
     1 |        1278
     2 |        1209
     3 |         830
     4 |        1404
     5 |         228
     6 |        1104
     7 |         908
     8 |         911
(9 rows)
```

5.List the total slots booked per facility in a given month

Query: select facid, sum(slots) as "Total Slots" from cd.bookings where starttime >= '2012-09-01' and starttime < '2012-10-01' group by facid order by sum(slots);

SQLOUTPUT:

```
exercises=# select facid, sum(slots) as "Total Slots" from cd.bookings where starttime >= '2012-09-01' and starttime < '2012-10-01' group by facid order by sum(slots);
 facid | Total Slots
-------+-------------
     5 |         122
     3 |         422
     7 |         426
     8 |         471
     6 |         540
     2 |         570
     1 |         588
     0 |         591
     4 |         648
(9 rows)
```

6.List the total slots booked per facility per month

Query: select facid, extract(month from starttime) as month, sum(slots) as "Total Slots" from cd.bookings where extract(year from starttime) = 2012 group by facid, month order by facid, month;

SQLOUTPUT:

```
exercises=# select facid, extract(month from starttime) as month, sum(slots) as "Total Slots" from cd.bookings where extract(year from starttime) = 2012 group by facid, month or
der by facid, month;
 facid | month | Total Slots
-------+-------+-------------
     0 |     7 |         270
     0 |     8 |         459
     0 |     9 |         591
     1 |     7 |         207
     1 |     8 |         483
     1 |     9 |         588
     2 |     7 |         180
     2 |     8 |         459
     2 |     9 |         570
     3 |     7 |         104
     3 |     8 |         304
     3 |     9 |         422
     4 |     7 |         264
     4 |     8 |         492
     4 |     9 |         648
     5 |     7 |          24
     5 |     8 |          82
     5 |     9 |         122
     6 |     7 |         164
     6 |     8 |         400
     6 |     9 |         540
     7 |     7 |         156
     7 |     8 |         326
     7 |     9 |         426
     8 |     7 |         117
     8 |     8 |         322
     8 |     9 |         471
(27 rows)
```

7.Find the count of members who have made at least one booking

Query: select count(*) from

(select distinct memid from cd.bookings) as mems;

SQLOUTPUT:

```
exercises=# select count(*) from
exercises-# (select distinct memid from cd.bookings) as mems
exercises-# ;
 count
-------
    30
(1 row)
```

8.List facilities with more than 1000 slots booked

Query: select facid, sum(slots) as "Total Slots" from cd.bookings group by facid having sum(slots) > 1000 order by facid;

SQLOUTPUT:

```
exercises=# select facid, sum(slots) as "Total Slots" from cd.bookings group by facid having sum(slots) > 1000 order by facid;
 facid | Total Slots
-------+-------------
     0 |        1320
     1 |        1278
     2 |        1209
     4 |        1404
     6 |        1104
(5 rows)
```

9.Find the total revenue of each facility

Query: select facs.name, sum(slots * case when memid = 0 then facs.guestcost else facs.membercost end) as revenue from cd.bookings bks inner join cd.facilities facs on bks.facid = facs.facid group by facs.name order by revenue;

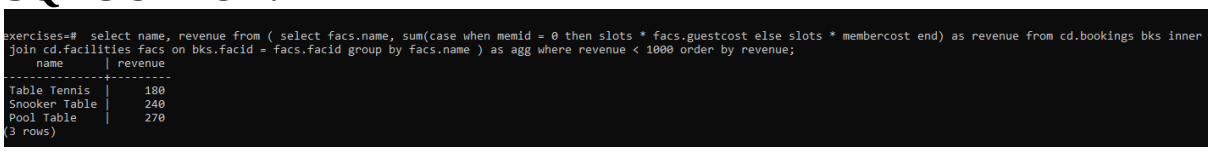
SQLOUTPUT:

```
exercises=# select facs.name, sum(slots * case when memid = 0 then facs.guestcost else facs.membercost end) as revenue from cd.bookings bks inner join cd.facilities facs on bks.
facid = facs.facid group by facs.name order by revenue;
      name       | revenue
-----------------+---------
 Table Tennis    |     180
 Snooker Table   |     240
 Pool Table      |     270
 Badminton Court |  1906.5
 Squash Court    | 13468.0
 Tennis Court 1  |   13860
 Tennis Court 2  |   14310
 Massage Room 2  |   15810
 Massage Room 1  |   72540
(9 rows)
```

10.Find facilities with a total revenue less than 1000

Query: select name, revenue from (select facs.name, sum(case when memid = 0 then slots * facs.guestcost else slots * membercost end) as revenue from cd.bookings bks inner join cd.facilities facs on bks.facid = facs.facid group by facs.name) as agg where revenue < 1000 order by revenue;

SQLOUTPUT:

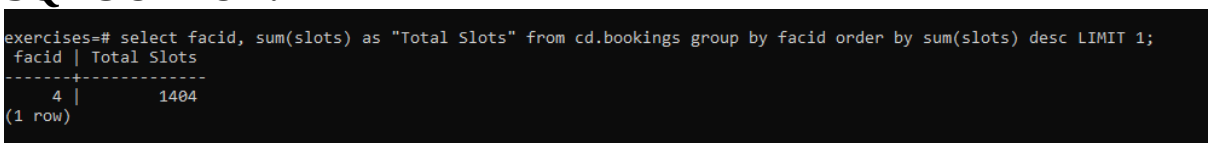


```
exercises=#  select name, revenue from ( select facs.name, sum(case when memid = 0 then slots * facs.guestcost else slots * membercost end) as revenue from cd.bookings bks inner
 join cd.facilities facs on bks.facid = facs.facid group by facs.name ) as agg where revenue < 1000 order by revenue;
    name      | revenue
--------------+---------
 Table Tennis |     180
 Snooker Table |     240
 Pool Table   |     270
(3 rows)
```

11.Output the facility id that has the highest number of slots booked

Query: select facid, sum(slots) as "Total Slots" from cd.bookings group by facid order by sum(slots) desc LIMIT 1;

SQLOUTPUT:



```
exercises=# select facid, sum(slots) as "Total Slots" from cd.bookings group by facid order by sum(slots) desc LIMIT 1;
 facid | Total Slots
-------+-------------
     4 |        1404
(1 row)
```

12.List the total slots booked per facility per month, part 2

Query: select facid, extract(month from starttime) as month, sum(slots) as slots from cd.bookings where starttime >= '2012-01-01' and starttime < '2013-01-01' group by facid, month union all select facid, null, sum(slots) as slots from cd.bookings where starttime >= '2012-01-01' and starttime < '2013-01-01' group by facid union all select null, null, sum(slots) as slots from cd.bookings where starttime >= '2012-01-01' and starttime < '2013-01-01' order by facid, month;

SQLOUTPUT:

```
exercises=# select facid, extract(month from starttime) as month, sum(slots) as slots from cd.bookings where starttime >= '2012-01-01' and starttime < '2013-01-01' group by faci
d, month union all select facid, null, sum(slots) as slots from cd.bookings where starttime >= '2012-01-01' and starttime < '2013-01-01' group by facid union all select null, nu
ll, sum(slots) as slots from cd.bookings where starttime >= '2012-01-01' and starttime < '2013-01-01' order by facid, month;
 facid | month | slots
-------+-------+-------
     0 |     7 |   270
     0 |     8 |   459
     0 |     9 |   591
     0 |       |  1320
     1 |     7 |   207
     1 |     8 |   483
     1 |     9 |   588
     1 |       |  1278
     2 |     7 |   180
     2 |     8 |   459
     2 |     9 |   570
     2 |       |  1209
     3 |     7 |   104
     3 |     8 |   304
     3 |     9 |   422
     3 |       |   830
     4 |     7 |   264
     4 |     8 |   492
     4 |     9 |   648
     4 |       |  1404
     5 |     7 |    24
     5 |     8 |    82
     5 |     9 |   122
     5 |       |   228
     6 |     7 |   164
     6 |     8 |   400
     6 |     9 |   540
     6 |       |  1104
     7 |     7 |   156
     7 |     8 |   326
     7 |     9 |   426
     7 |       |   908
     8 |     7 |   117
     8 |     8 |   322
     8 |     9 |   471
     8 |       |   910
       |       |  9191
(37 rows)
```

13.List the total hours booked per named facility

Query: select facs.facid, facs.name, trim(to_char(sum(bks.slots)/2.0, '9999999999999999D99')) as "Total Hours" from cd.bookings bks inner join cd.facilities facs on facs.facid = bks.facid group by facs.facid, facs.name order by facs.facid;

SQLOUTPUT:

```
exercises=# select facs.facid, facs.name, trim(to_char(sum(bks.slots)/2.0, '9999999999999999D99')) as "Total Hours" from cd.bookings bks inner join cd.facilities facs on facs.fa
cid = bks.facid group by facs.facid, facs.name order by facs.facid;
 facid |      name       | Total Hours
-------+-----------------+-------------
     0 | Tennis Court 1  | 660.00
     1 | Tennis Court 2  | 639.00
     2 | Badminton Court | 604.50
     3 | Table Tennis    | 415.00
     4 | Massage Room 1  | 702.00
     5 | Massage Room 2  | 114.00
     6 | Squash Court    | 552.00
     7 | Snooker Table   | 454.00
     8 | Pool Table      | 455.50
(9 rows)
```

14.List each member's first booking after September 1st 2012

Query: select mems.surname, mems.firstname, mems.memid, min(bks.starttime) as starttime from cd.bookings bks inner join cd.members mems on mems.memid = bks.memid where starttime >= '2012-09-01' group by mems.surname, mems.firstname, mems.memid order by mems.memid;

SQLOUTPUT:

```
exercises=# select mems.surname, mems.firstname, mems.memid, min(bks.starttime) as starttime from cd.bookings bks inner join cd.members mems on mems.memid = bks.memid where star
ttime >= '2012-09-01' group by mems.surname, mems.firstname, mems.memid order by mems.memid;
      surname      | firstname | memid |      starttime
-------------------+-----------+-------+---------------------
 GUEST             | GUEST     |     0 | 2012-09-01 08:00:00
 Smith             | Darren    |     1 | 2012-09-01 09:00:00
 Smith             | Tracy     |     2 | 2012-09-01 11:30:00
 Rownam            | Tim       |     3 | 2012-09-01 16:00:00
 Joplette          | Janice    |     4 | 2012-09-01 15:00:00
 Butters           | Gerald    |     5 | 2012-09-02 12:30:00
 Tracy             | Burton    |     6 | 2012-09-01 15:00:00
 Dare              | Nancy     |     7 | 2012-09-01 12:30:00
 Boothe            | Tim       |     8 | 2012-09-01 08:30:00
 Stibbons          | Ponder    |     9 | 2012-09-01 11:00:00
 Owen              | Charles   |    10 | 2012-09-01 11:00:00
 Jones             | David     |    11 | 2012-09-01 09:30:00
 Baker             | Anne      |    12 | 2012-09-01 14:30:00
 Farrell           | Jemima    |    13 | 2012-09-01 09:30:00
 Smith             | Jack      |    14 | 2012-09-01 11:00:00
 Bader             | Florence  |    15 | 2012-09-01 10:30:00
 Baker             | Timothy   |    16 | 2012-09-01 15:00:00
 Pinker            | David     |    17 | 2012-09-01 08:30:00
 Genting           | Matthew   |    20 | 2012-09-01 18:00:00
 Mackenzie         | Anna      |    21 | 2012-09-01 08:30:00
 Coplin            | Joan      |    22 | 2012-09-02 11:30:00
 Sarwin            | Ramnaresh |    24 | 2012-09-04 11:00:00
 Jones             | Douglas   |    26 | 2012-09-08 13:00:00
 Rumney            | Henrietta |    27 | 2012-09-16 13:30:00
 Farrell           | David     |    28 | 2012-09-18 09:00:00
 Worthington-Smyth | Henry     |    29 | 2012-09-19 09:30:00
 Purview           | Millicent |    30 | 2012-09-19 11:30:00
 Tupperware        | Hyacinth  |    33 | 2012-09-20 08:00:00
 Hunt              | John      |    35 | 2012-09-23 14:00:00
 Crumpet           | Erica     |    36 | 2012-09-27 11:30:00
(30 rows)
```

15.Produce a list of member names, with each row containing the total member count

Query: select (select count(*) from cd.members) as count, firstname, surname from cd.members order by joindate;

SQLOUTPUT:

```
exercises=# select (select count(*) from cd.members) as count, firstname, surname from cd.members order by joindate;
 count | firstname |      surname
-------+-----------+-------------------
    31 | GUEST     | GUEST
    31 | Darren    | Smith
    31 | Tracy     | Smith
    31 | Tim       | Rownam
    31 | Janice    | Joplette
    31 | Gerald    | Butters
    31 | Burton    | Tracy
    31 | Nancy     | Dare
    31 | Tim       | Boothe
    31 | Ponder    | Stibbons
    31 | Charles   | Owen
    31 | David     | Jones
    31 | Anne      | Baker
    31 | Jemima    | Farrell
    31 | Jack      | Smith
    31 | Florence  | Bader
    31 | Timothy   | Baker
    31 | David     | Pinker
    31 | Matthew   | Genting
    31 | Anna      | Mackenzie
    31 | Joan      | Coplin
    31 | Ramnaresh | Sarwin
    31 | Douglas   | Jones
    31 | Henrietta | Rumney
    31 | David     | Farrell
    31 | Henry     | Worthington-Smyth
    31 | Millicent | Purview
    31 | Hyacinth  | Tupperware
    31 | John      | Hunt
    31 | Erica     | Crumpet
    31 | Darren    | Smith
(31 rows)
```

16.Produce a numbered list of members

Query: select row_number() over(order by joindate), firstname, surname from cd.members order by joindate;

SQLOUTPUT:

```
exercises=# select row_number() over(order by joindate), firstname, surname from cd.members order by joindate;
 row_number | firstname |      surname
------------+-----------+-------------------
          1 | GUEST     | GUEST
          2 | Darren    | Smith
          3 | Tracy     | Smith
          4 | Tim       | Rownam
          5 | Janice    | Joplette
          6 | Gerald    | Butters
          7 | Burton    | Tracy
          8 | Nancy     | Dare
          9 | Tim       | Boothe
         10 | Ponder    | Stibbons
         11 | Charles   | Owen
         12 | David     | Jones
         13 | Anne      | Baker
         14 | Jemima    | Farrell
         15 | Jack      | Smith
         16 | Florence  | Bader
         17 | Timothy   | Baker
         18 | David     | Pinker
         19 | Matthew   | Genting
         20 | Anna      | Mackenzie
         21 | Joan      | Coplin
         22 | Ramnaresh | Sarwin
         23 | Douglas   | Jones
         24 | Henrietta | Rumney
         25 | David     | Farrell
         26 | Henry     | Worthington-Smyth
         27 | Millicent | Purview
         28 | Hyacinth  | Tupperware
         29 | John      | Hunt
         30 | Erica     | Crumpet
         31 | Darren    | Smith
(31 rows)
```

17.Output the facility id that has the highest number of slots booked, again

Query:

select facid, sum(slots) as totalslots from cd.bookings group by facid having sum(slots) = (select max(sum2.totalslots) from (select sum(slots) as totalslots from cd.bookings group by facid) as sum2);

SQLOUTPUT:

```
exercises=# select facid, sum(slots) as totalslots from cd.bookings group by facid having sum(slots) = (select max(sum2.totalslots) from (select sum(slots) as totalslots from cd
.bookings group by facid ) as sum2);
 facid | totalslots
-------+------------
     4 |       1404
(1 row)
```

18.Rank members by (rounded) hours used

Query: select firstname, surname, hours, rank() over (order by hours desc) from (select firstname, surname, ((sum(bks.slots)+10)/20)*10 as hours from cd.bookings bks inner join cd.members mems on bks.memid = mems.memid group by mems.memid) as subq order by rank, surname, firstname;

SQLOUTPUT:

```
exercises=# select firstname, surname, hours, rank() over (order by hours desc) from (select firstname, surname, ((sum(bks.slots)+10)/20)*10 as hours from cd.bookings bks inner
join cd.members mems on bks.memid = mems.memid group by mems.memid ) as subq order by rank, surname, firstname;
 firstname |      surname      | hours | rank
-----------+-------------------+-------+------
 GUEST     | GUEST             |  1200 |    1
 Darren    | Smith             |   340 |    2
 Tim       | Rownam            |   330 |    3
 Tim       | Boothe            |   220 |    4
 Tracy     | Smith             |   220 |    4
 Gerald    | Butters           |   210 |    6
 Burton    | Tracy             |   180 |    7
 Charles   | Owen              |   170 |    8
 Janice    | Joplette          |   160 |    9
 Anne      | Baker             |   150 |   10
 Timothy   | Baker             |   150 |   10
 David     | Jones             |   150 |   10
 Nancy     | Dare              |   130 |   13
 Florence  | Bader             |   120 |   14
 Anna      | Mackenzie         |   120 |   14
 Ponder    | Stibbons          |   120 |   14
 Jack      | Smith             |   110 |   17
 Jemima    | Farrell           |    90 |   18
 David     | Pinker            |    80 |   19
 Ramnaresh | Sarwin            |    80 |   19
 Matthew   | Genting           |    70 |   21
 Joan      | Coplin            |    50 |   22
 David     | Farrell           |    30 |   23
 Henry     | Worthington-Smyth |    30 |   23
 John      | Hunt              |    20 |   25
 Douglas   | Jones             |    20 |   25
 Millicent | Purview           |    20 |   25
 Henrietta | Rumney            |    20 |   25
 Erica     | Crumpet           |    10 |   29
 Hyacinth  | Tupperware        |    10 |   29
(30 rows)
```

19.Find the top three revenue generating facilities

Query: select name, rank from (select facs.name as name, rank() over (order by sum(case when memid = 0 then slots * facs.guestcost else slots * membercost end) desc) as rank from cd.bookings bks inner join cd.facilities facs on bks.facid = facs.facid group by facs.name) as subq where rank <= 3 order by rank;

SQLOUTPUT:

```
exercises=#  select name, rank from ( select facs.name as name, rank() over (order by sum(case when memid = 0 then slots * facs.guestcost else slots * membercost end) desc) as r
ank from cd.bookings bks inner join cd.facilities facs on bks.facid = facs.facid group by facs.name ) as subq where rank <= 3 order by rank;
      name      | rank
----------------+------
 Massage Room 1 |    1
 Massage Room 2 |    2
 Tennis Court 2 |    3
(3 rows)
```

20.Classify facilities by value

Query: select name, case when class=1 then 'high' when class=2 then 'average' else 'low' end revenue from (select facs.name as name, ntile(3) over (order by sum(case when memid = 0 then slots * facs.guestcost else slots * membercost end) desc) as class from cd.bookings bks inner join cd.facilities facs on bks.facid = facs.facid group by facs.name) as subq order by class, name;

SQLOUTPUT:

```
exercises=# select name, case when class=1 then 'high' when class=2 then 'average' else 'low' end revenue from ( select facs.name as name, ntile(3) over (order by sum(case when
memid = 0 then slots * facs.guestcost else slots * membercost end) desc) as class from cd.bookings bks inner join cd.facilities facs on bks.facid = facs.facid group by facs.name
 ) as subq order by class, name;
      name       | revenue
-----------------+---------
 Massage Room 1  | high
 Massage Room 2  | high
 Tennis Court 2  | high
 Badminton Court | average
 Squash Court    | average
 Tennis Court 1  | average
 Pool Table      | low
 Snooker Table   | low
 Table Tennis    | low
(9 rows)
```

21.Calculate the payback time for each facility

Query: select name, initialoutlay / (monthlyrevenue - monthlymaintenance) as repaytime from (select facs.name as name, facs.initialoutlay as initialoutlay, facs.monthlymaintenance as monthlymaintenance, sum(case when memid = 0 then slots * facs.guestcost else slots * membercost end)/3 as monthlyrevenue from cd.bookings bks inner join cd.facilities facs on bks.facid = facs.facid group by facs.facid) as subq order by name;

SQLOUTPUT:

```
exercises=# select name, initialoutlay / (monthlyrevenue - monthlymaintenance) as repaytime from (select facs.name as name, facs.initialoutlay as initialoutlay, facs.monthlymain
tenance as monthlymaintenance, sum(case when memid = 0 then slots * facs.guestcost else slots * membercost end)/3 as monthlyrevenue from cd.bookings bks inner join cd.facilities
 facs on bks.facid = facs.facid group by facs.facid ) as subq order by name;
      name       |       repaytime
-----------------+------------------------
 Badminton Court |     6.8317677198975235
 Massage Room 1  | 0.18885741265344664778
 Massage Room 2  |     1.7621145374449339
 Pool Table      |     5.3333333333333333
 Snooker Table   |     6.9230769230769231
 Squash Court    |     1.1339582703356516
 Table Tennis    |     6.4000000000000000
 Tennis Court 1  |     2.2624434389140271
 Tennis Court 2  |     1.7505470459518600
(9 rows)
```

22.Calculate a rolling average of total revenue

Query: select dategen.date, (-- *correlated subquery that, for each day fed into it, -- finds the average revenue for the last 15 days*

select sum(case when memid = 0 then slots * facs.guestcost else slots * membercost end)

as rev from cd.bookings bks inner join cd.facilities facs on bks.facid = facs.facid where bks.starttime > dategen.date - interval '14 days' and bks.starttime < dategen.date + interval '1 day')/15 as revenue

from (-- *generates a list of days in august* select cast(generate_series(timestamp '2012-08-01', '2012-08-31','1 day') as date) as date) as dategen order by dategen.date;

SQLOUTPUT:

```
exercises=# select dategen.date,
exercises-# (
exercises(# -- correlated subquery that, for each day fed into it,
exercises(# -- finds the average revenue for the last 15 days
exercises(# select sum(case
exercises(# when memid = 0 then slots * facs.guestcost
exercises(# else slots * membercost
exercises(# end) as rev
exercises(#
exercises(# from cd.bookings bks
exercises(# inner join cd.facilities facs
exercises(# on bks.facid = facs.facid
exercises(# where bks.starttime > dategen.date - interval '14 days'
exercises(# and bks.starttime < dategen.date + interval '1 day'
exercises(# )/15 as revenue
exercises-# from
exercises-# (
exercises(# -- generates a list of days in august
exercises(# select cast(generate_series(timestamp '2012-08-01',
exercises(# '2012-08-31','1 day') as date) as date
exercises(# )  as dategen
exercises-# order by dategen.date;
    date    |        revenue
------------+-----------------------
 2012-08-01 | 1126.8333333333333333
 2012-08-02 | 1153.0000000000000000
 2012-08-03 | 1162.9000000000000000
 2012-08-04 | 1177.3666666666666667
 2012-08-05 | 1160.9333333333333333
 2012-08-06 | 1185.4000000000000000
 2012-08-07 | 1182.8666666666666667
 2012-08-08 | 1172.6000000000000000
 2012-08-09 | 1152.4666666666666667
 2012-08-10 | 1175.0333333333333333
 2012-08-11 | 1176.6333333333333333
 2012-08-12 | 1195.6666666666666667
 2012-08-13 | 1218.0000000000000000
 2012-08-14 | 1247.4666666666666667
 2012-08-15 | 1274.1000000000000000
 2012-08-16 | 1281.2333333333333333
 2012-08-17 | 1324.4666666666666667
 2012-08-18 | 1373.7333333333333333
 2012-08-19 | 1406.0666666666666667
 2012-08-20 | 1427.0666666666666667
 2012-08-21 | 1450.3333333333333333
 2012-08-22 | 1539.7000000000000000
 2012-08-23 | 1567.3000000000000000
```

```
exercises(# end) as rev
exercises(#
exercises(# from cd.bookings bks
exercises(# inner join cd.facilities facs
exercises(# on bks.facid = facs.facid
exercises(# where bks.starttime > dategen.date - interval '14 days'
exercises(# and bks.starttime < dategen.date + interval '1 day'
exercises(# )/15 as revenue
exercises-# from
exercises-# (
exercises(# -- generates a list of days in august
exercises(# select cast(generate_series(timestamp '2012-08-01',
exercises(# '2012-08-31','1 day') as date) as date
exercises(# )  as dategen
exercises-# order by dategen.date;
    date    |        revenue
------------+-----------------------
 2012-08-01 | 1126.8333333333333333
 2012-08-02 | 1153.0000000000000000
 2012-08-03 | 1162.9000000000000000
 2012-08-04 | 1177.3666666666666667
 2012-08-05 | 1160.9333333333333333
 2012-08-06 | 1185.4000000000000000
 2012-08-07 | 1182.8666666666666667
 2012-08-08 | 1172.6000000000000000
 2012-08-09 | 1152.4666666666666667
 2012-08-10 | 1175.0333333333333333
 2012-08-11 | 1176.6333333333333333
 2012-08-12 | 1195.6666666666666667
 2012-08-13 | 1218.0000000000000000
 2012-08-14 | 1247.4666666666666667
 2012-08-15 | 1274.1000000000000000
 2012-08-16 | 1281.2333333333333333
 2012-08-17 | 1324.4666666666666667
 2012-08-18 | 1373.7333333333333333
 2012-08-19 | 1406.0666666666666667
 2012-08-20 | 1427.0666666666666667
 2012-08-21 | 1450.3333333333333333
 2012-08-22 | 1539.7000000000000000
 2012-08-23 | 1567.3000000000000000
 2012-08-24 | 1592.3333333333333333
 2012-08-25 | 1615.0333333333333333
 2012-08-26 | 1631.2000000000000000
 2012-08-27 | 1659.4333333333333333
 2012-08-28 | 1687.0000000000000000
 2012-08-29 | 1684.6333333333333333
 2012-08-30 | 1657.9333333333333333
 2012-08-31 | 1703.4000000000000000
(31 rows)
```