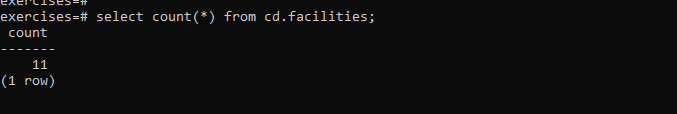
**Aggregation Task**

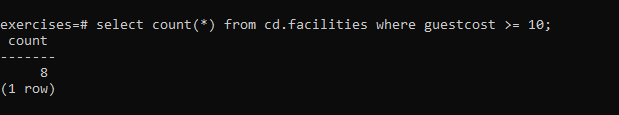
1. [**Count the number of facilities**](https://pgexercises.com/questions/aggregates/count.html)

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

**OUTPUT: **

**2.[Count the number of expensive facilities](https://pgexercises.com/questions/aggregates/count2.html)**

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

**OUTPUT: **

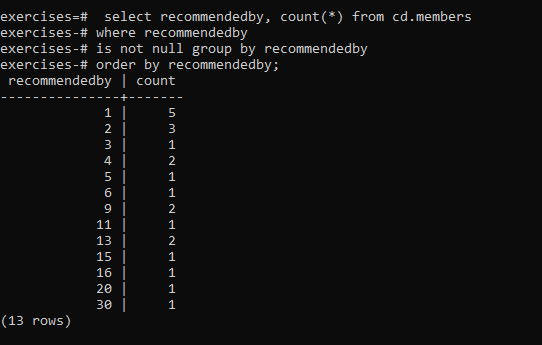
**3.[Count the number of recommendations each member makes.](https://pgexercises.com/questions/aggregates/count3.html)**

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

where recommendedby

is not null group by recommendedby

order by recommendedby;

**OUTPUT: **

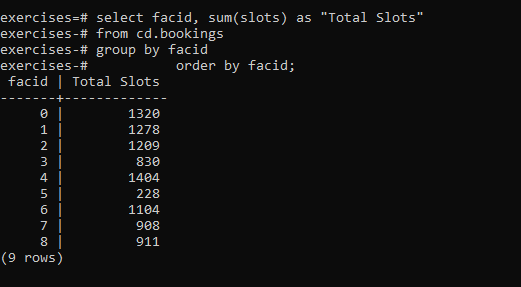
**4.[List the total slots booked per facility](https://pgexercises.com/questions/aggregates/fachours.html)**

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

from cd.bookings

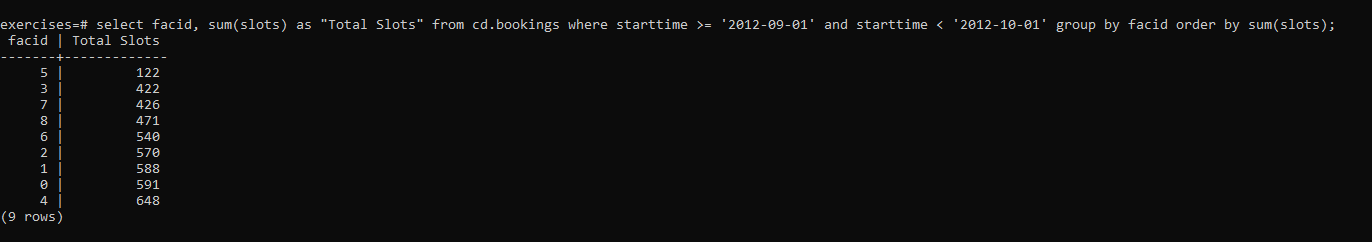
group by facid

order by facid;

**OUTPUT: **

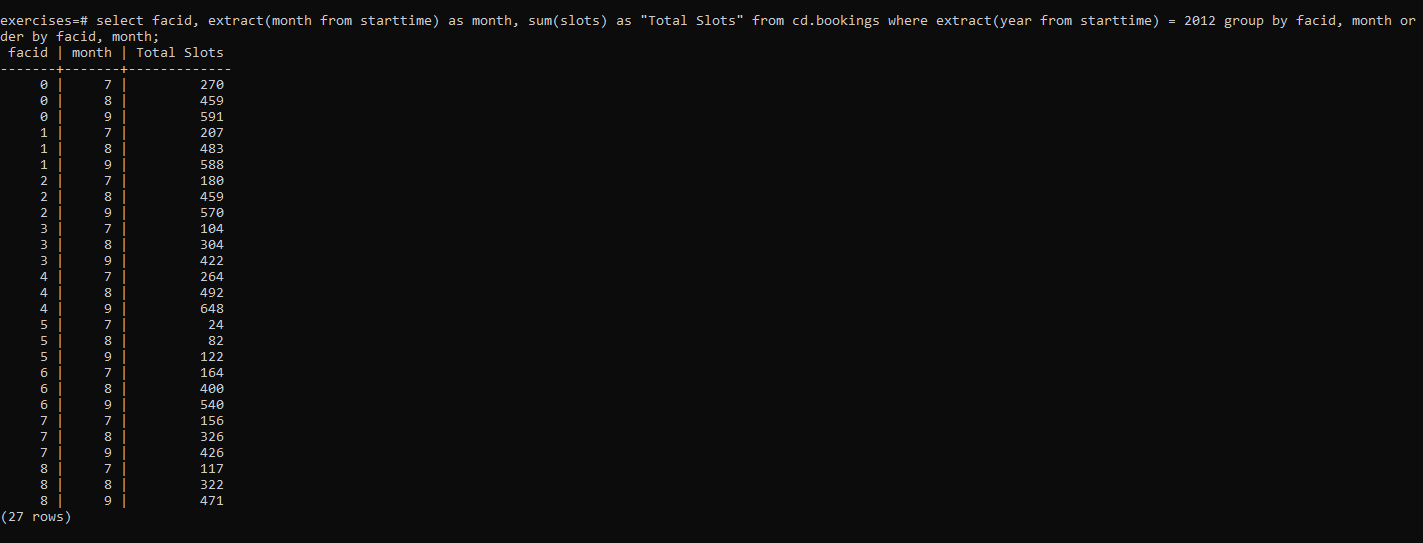
**5.[List the total slots booked per facility in a given month](https://pgexercises.com/questions/aggregates/fachoursbymonth.html)**

**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);

**OUTPUT: **

**6.[List the total slots booked per facility per month](https://pgexercises.com/questions/aggregates/fachoursbymonth2.html)**

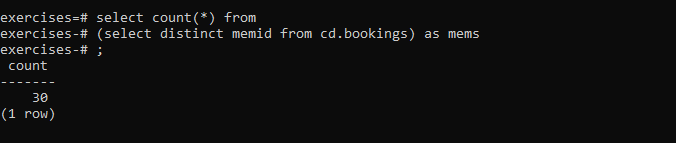
**Query:** select facid, extract(month from starttime) as month, sum(slots) as "Total Slots" from cd.bookings where extract(year from starttime) = 2012 group by facid, month order by facid, month;

**OUTPUT: **

**7.[Find the count of members who have made at least one booking](https://pgexercises.com/questions/aggregates/members1.html)**

**Query:** select count(\*) from

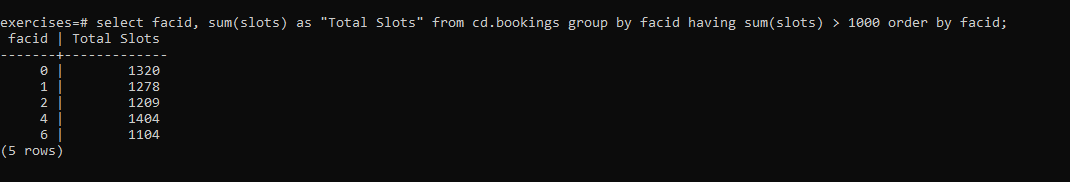
(select distinct memid from cd.bookings) as mems;

**OUTPUT: **

**8.[List facilities with more than 1000 slots booked](https://pgexercises.com/questions/aggregates/fachours1a.html)**

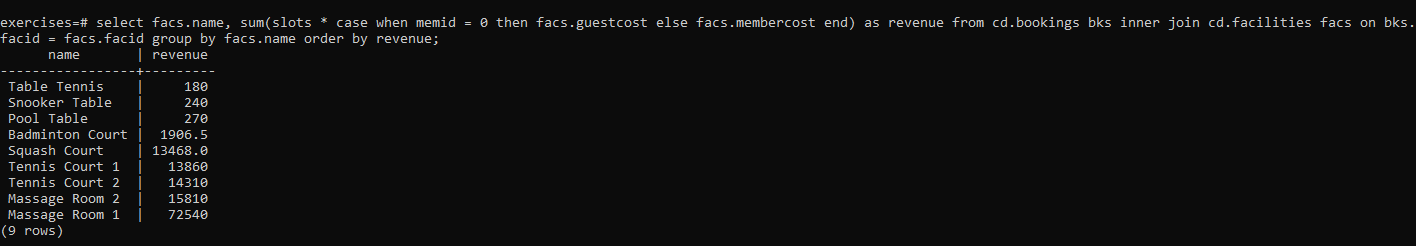
**Query:** select facid, sum(slots) as "Total Slots" from cd.bookings group by facid having sum(slots) > 1000 order by facid;

**OUTPUT:**

****

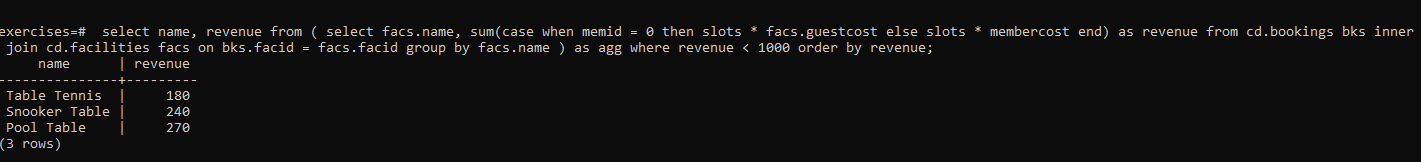
**9.[Find the total revenue of each facility](https://pgexercises.com/questions/aggregates/facrev.html)**

**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;

**OUTPUT: **

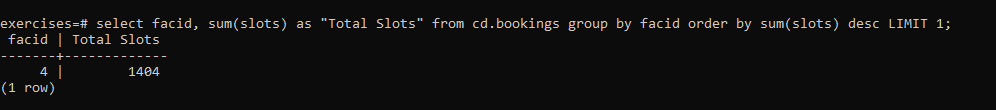
**10.[Find facilities with a total revenue less than 1000](https://pgexercises.com/questions/aggregates/facrev2.html)**

**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;

**OUTPUT: **

**11.[Output the facility id that has the highest number of slots booked](https://pgexercises.com/questions/aggregates/fachours2.html)**

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

**OUTPUT: **

**12.[List the total slots booked per facility per month, part 2](https://pgexercises.com/questions/aggregates/fachoursbymonth3.html)**

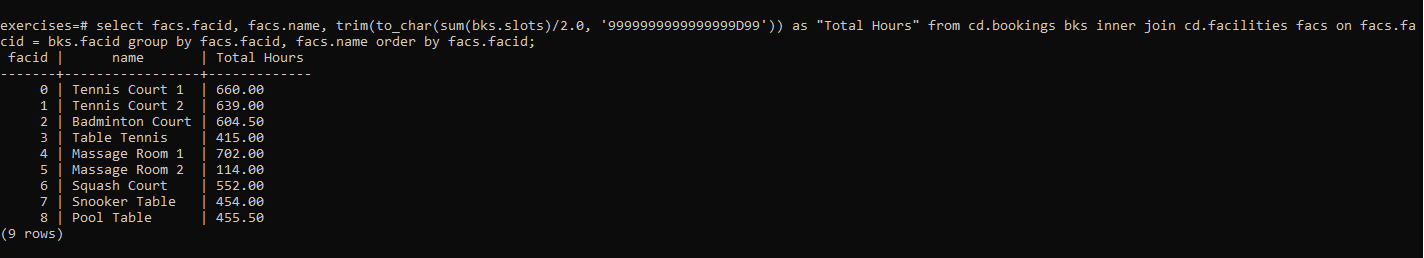
**Query:** select facid, extract(month from starttime) as month, sum(slots) as slots from cd.bookings where starttime >= '2012-01-01' and starttime < '2013-01-01' group by facid, month union all select facid, null, sum(slots) as slots from cd.bookings where starttime >= '2012-01-01' and starttime < '2013-01-01'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

**13.[List the total hours booked per named facility](https://pgexercises.com/questions/aggregates/fachours3.html)**

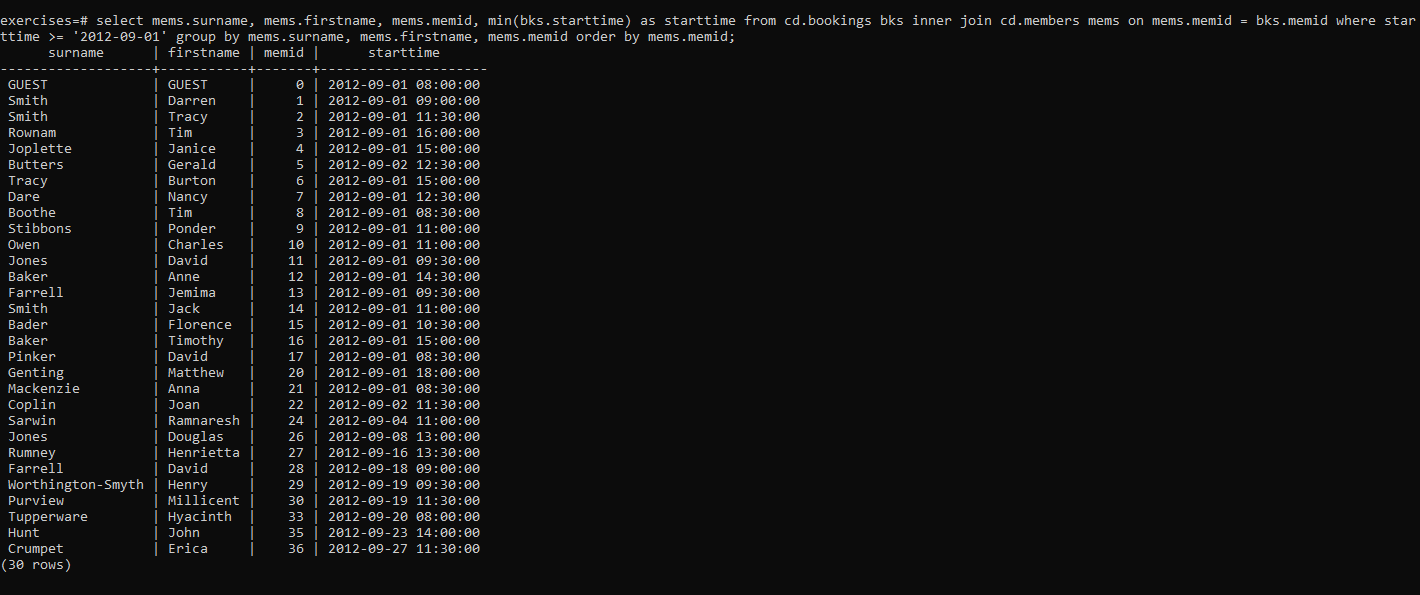
**Query:** select facs.facid, facs.name, trim(to\_char(sum(bks.slots)/2.0, '9999999999999999D99')) as "Total Hours" from cd.bookings bks inner join cd.facilities facs on facs.facid = bks.facid group by facs.facid, facs.name order by facs.facid;

**OUTPUT:**

****

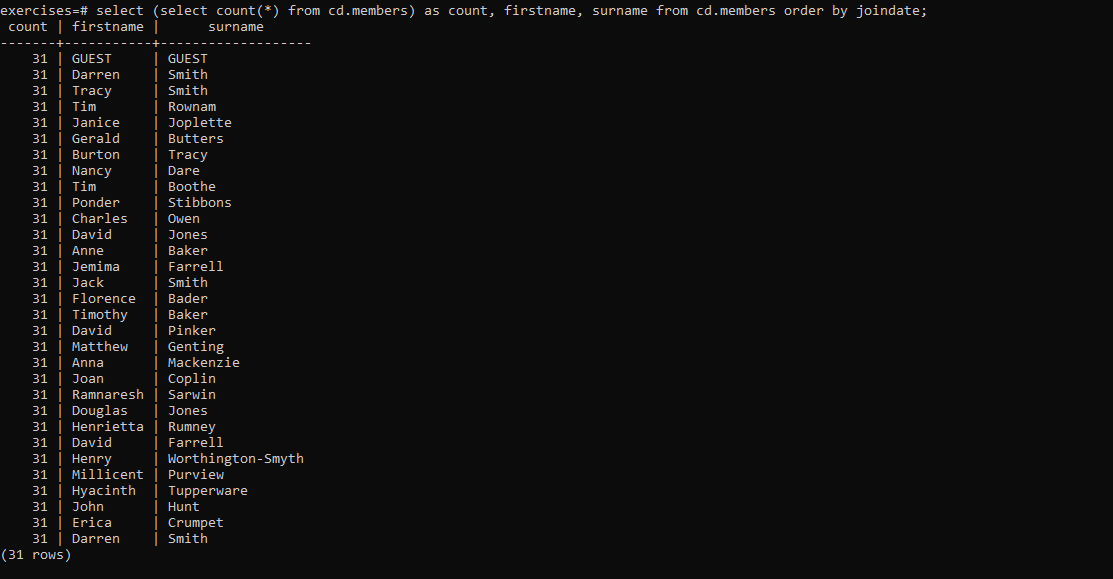
**14.[List each member's first booking after September 1st 2012](https://pgexercises.com/questions/aggregates/nbooking.html)**

**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;

**OUTPUT: **

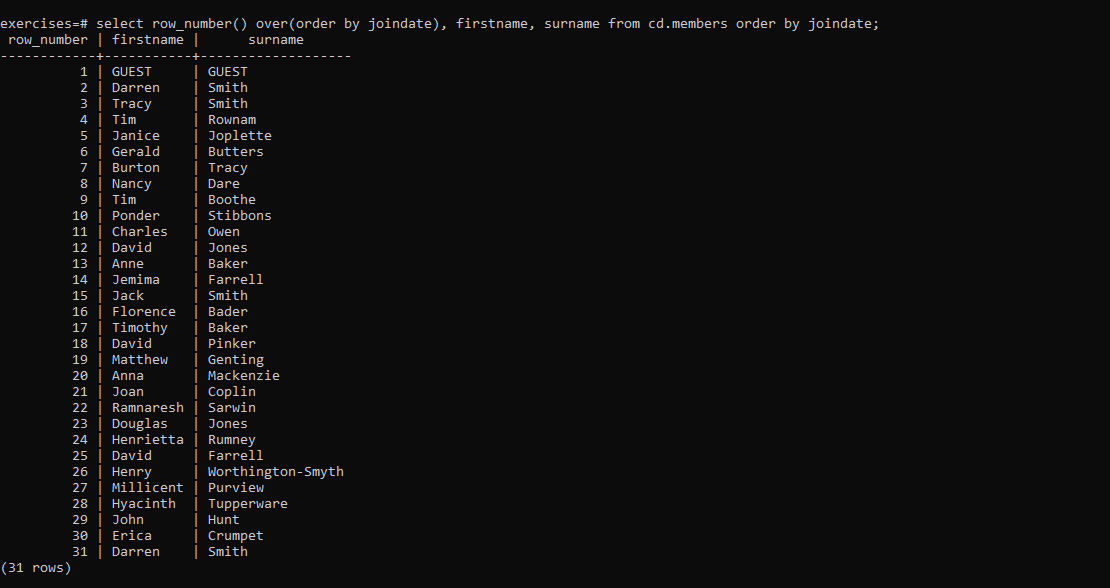
**15.[Produce a list of member names, with each row containing the total member count](https://pgexercises.com/questions/aggregates/countmembers.html)**

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

**OUTPUT: **

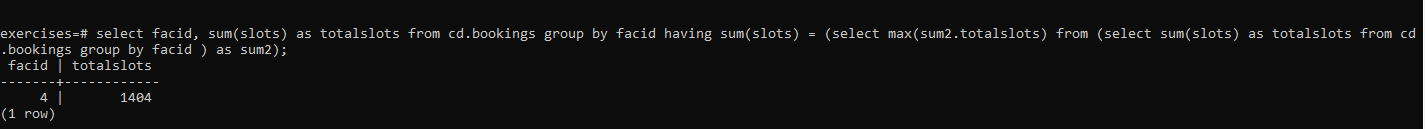
**16.[Produce a numbered list of members](https://pgexercises.com/questions/aggregates/nummembers.html)**

**Query:** select row\_number() over(order by joindate), firstname, surname from cd.members order by joindate;

**OUTPUT: **

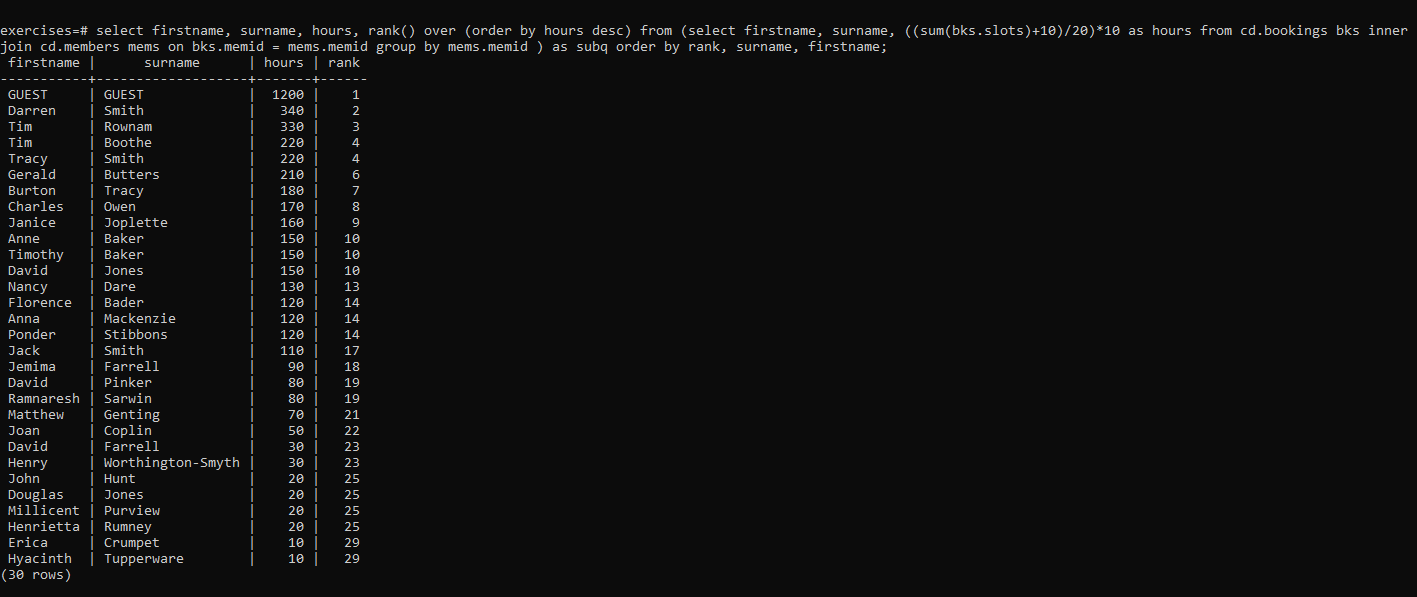
**17.[Output the facility id that has the highest number of slots booked, again](https://pgexercises.com/questions/aggregates/fachours4.html)**

**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);

**OUTPUT: **

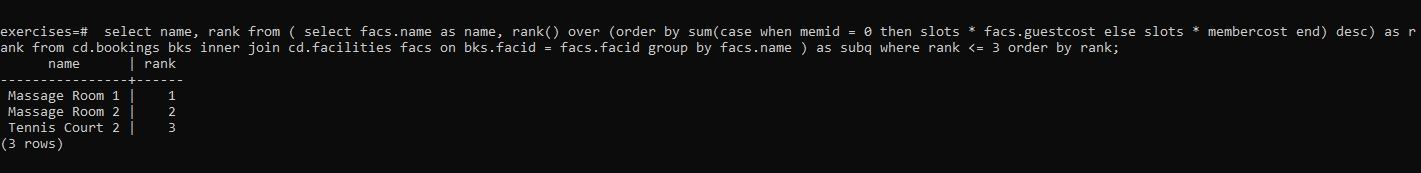
**18.[Rank members by (rounded) hours used](https://pgexercises.com/questions/aggregates/rankmembers.html)**

**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;

**OUTPUT: **

**19.[Find the top three revenue generating facilities](https://pgexercises.com/questions/aggregates/facrev3.html)**

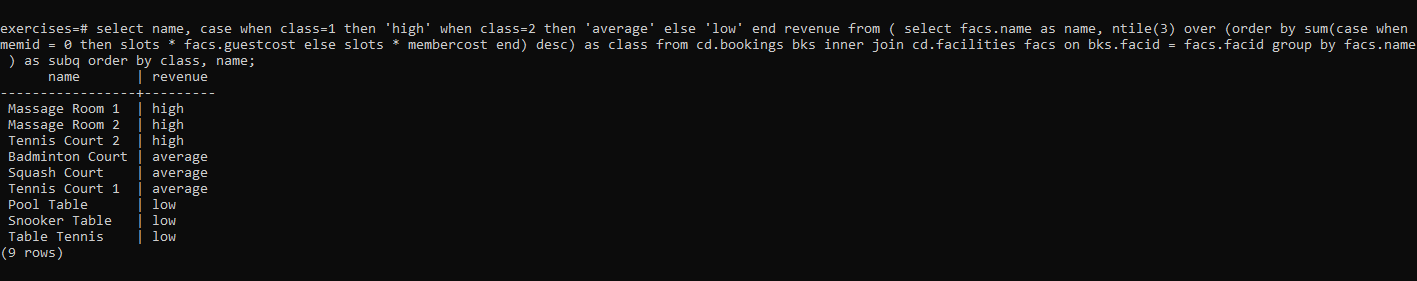
**Query:** select name, rank from ( select facs.name as name, rank() over (order by sum(case when memid = 0 then slots \* facs.guestcost else slots \* membercost end) desc) as rank from cd.bookings bks inner join cd.facilities facs on bks.facid = facs.facid group by facs.name ) as subq where rank <=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

**OUTPUT: **

**20.[Classify facilities by value](https://pgexercises.com/questions/aggregates/classify.html)**

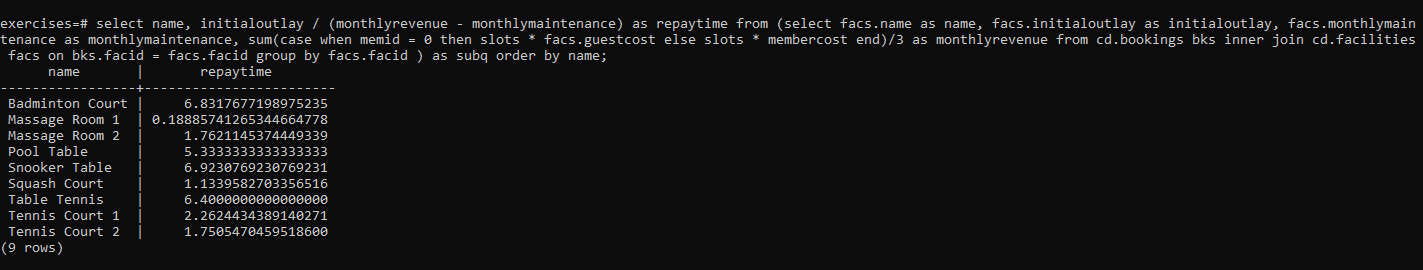
**Query:** select name, case when class=1 then 'high' when class=2 then 'average' else 'low' end revenue from ( select facs.name as name, ntile(3) over (order by sum(case when memid = 0 then slots \* facs.guestcost else slots \* membercost end) desc) as class from cd.bookings bks inner join cd.facilities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

**OUTPUT:**

****

**21.[Calculate the payback time for each facility](https://pgexercises.com/questions/aggregates/payback.html)**

**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;

**OUTPUT: **

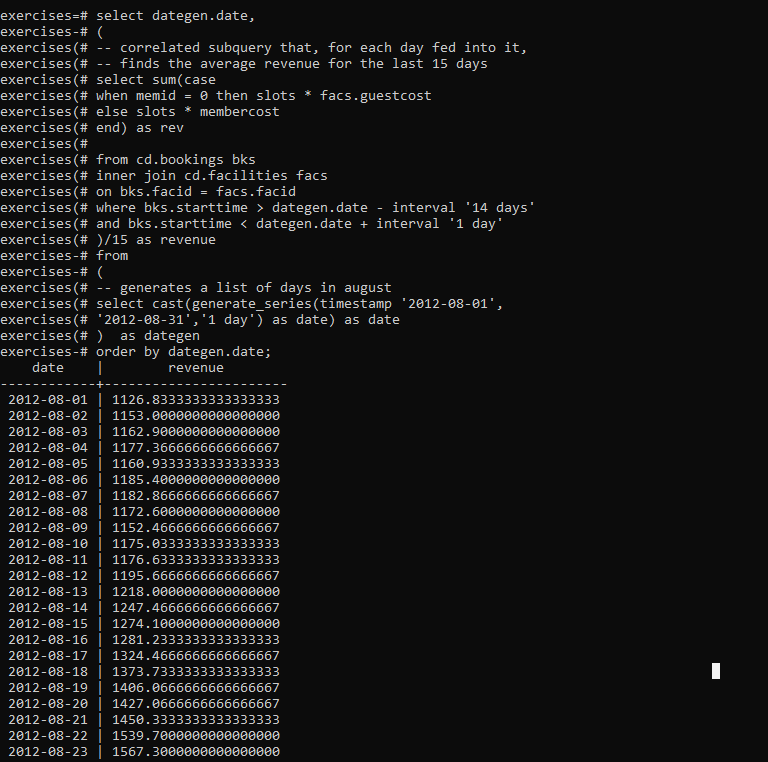
**22.[Calculate a rolling average of total revenue](https://pgexercises.com/questions/aggregates/rollingavg.html)**

**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;

**OUTPUT: **

