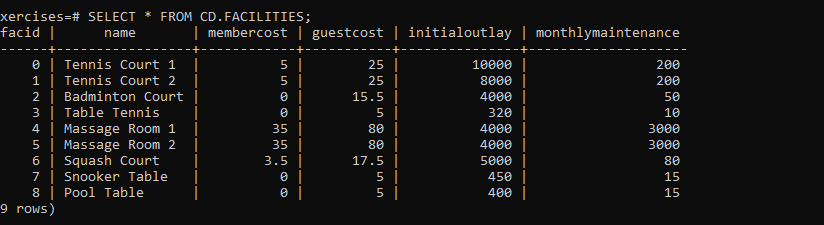
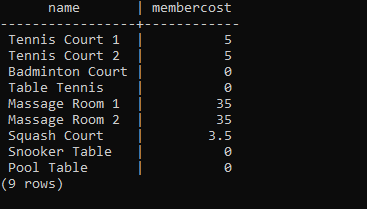
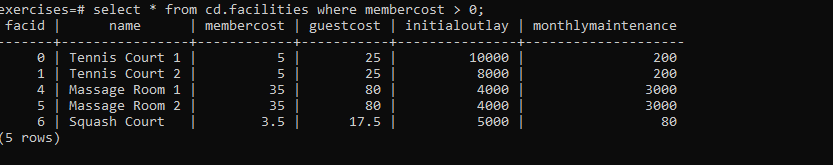
# //Category1 :Basic

**1.How can you retrieve all the information from the cd.facilities table?**

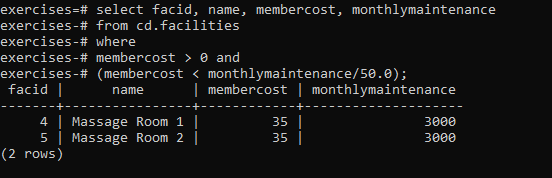
  
**2.You want to print out a list of all of the facilities and their cost to members. How would you retrieve a list of only facility names and costs?**



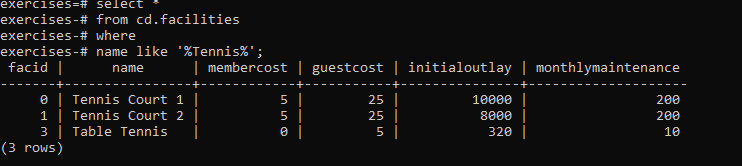
**3.How can you produce a list of facilities that charge a fee to members?**



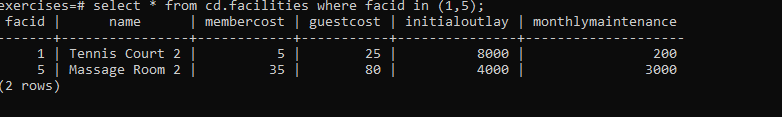
**4.How can you produce a list of facilities that charge a fee to members, and that fee is less than 1/50th of the monthly maintenance cost? Return the facid, facility name, member cost, and monthly maintenance of the facilities in question.**



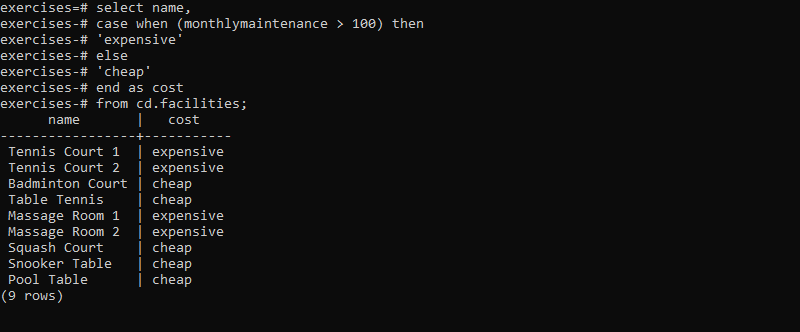
**5.How can you produce a list of all facilities with the word 'Tennis' in their name?**



**6. How can you retrieve the details of facilities with ID 1 and 5? Try to do it without using the OR operator.**



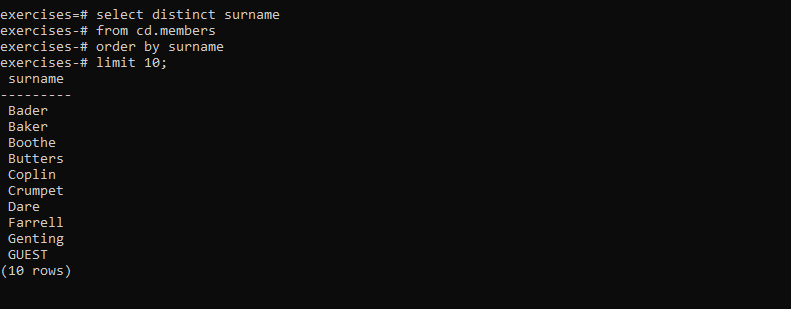
**7. How can you produce a list of facilities, with each labelled as 'cheap' or 'expensive' depending on if their monthly maintenance cost is more than $100? Return the name and monthly maintenance of the facilities in question.**



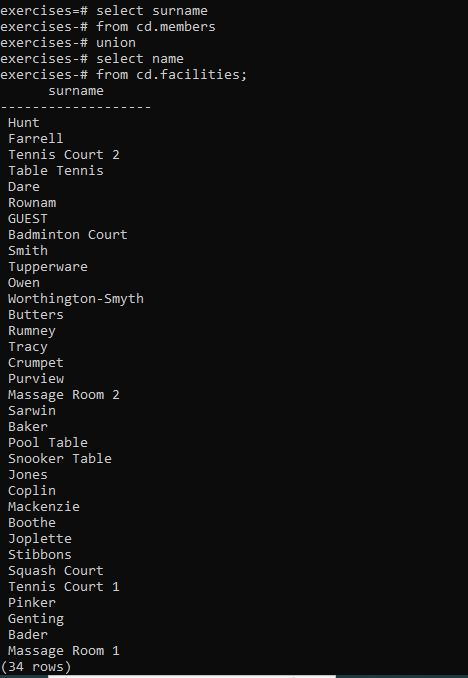
**8.** **How can you produce a list of members who joined after the start of September 2012? Return the memid, surname, firstname, and joindate of the members in question.**



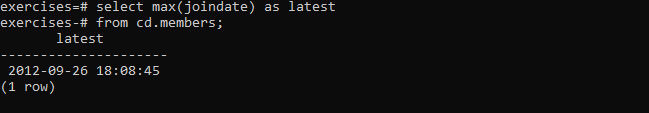
**9. How can you produce an ordered list of the first 10 surnames in the members table? The list must not contain duplicates.**



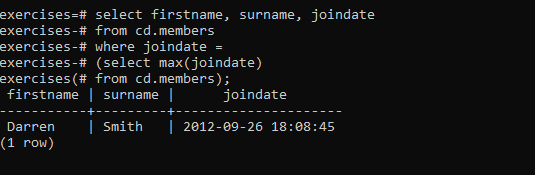
**10.** **You, for some reason, want a combined list of all surnames and all facility names. Yes, this is a contrived example :-). Produce that list!**



**11.** **You'd like to get the signup date of your last member. How can you retrieve this information?**

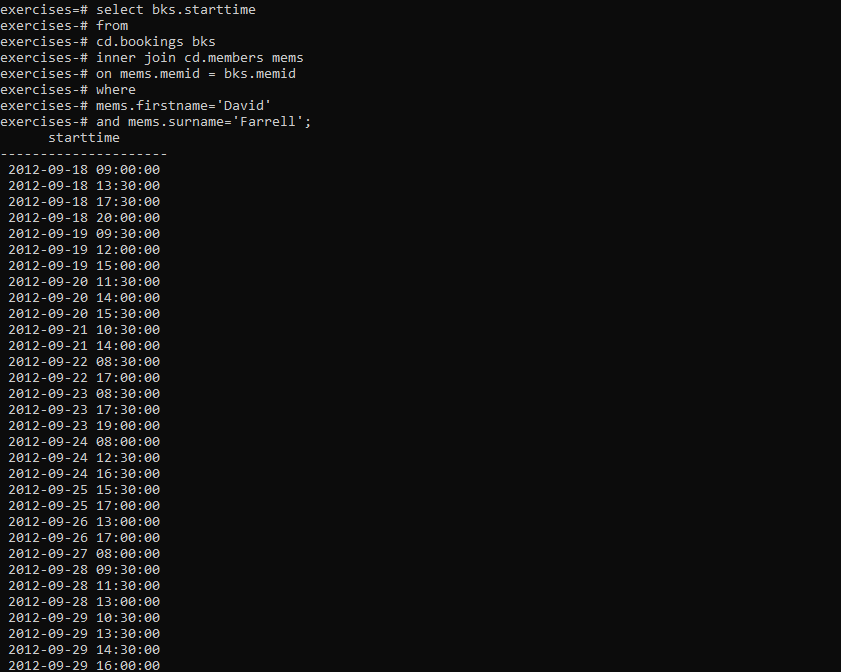


**12. You'd like to get the first and last name of the last member(s) who signed up - not just the date. How can you do that?**

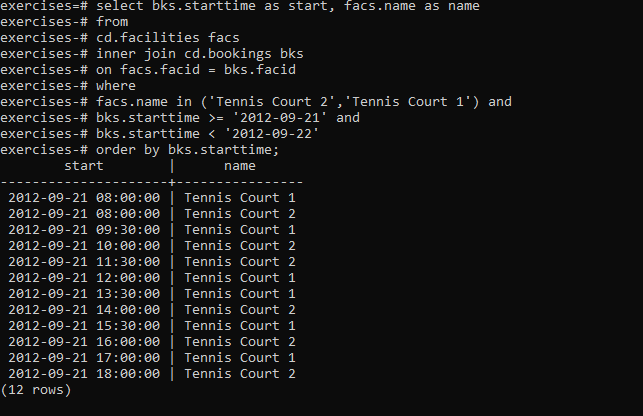


# //category -2 :Joins and Subqueries

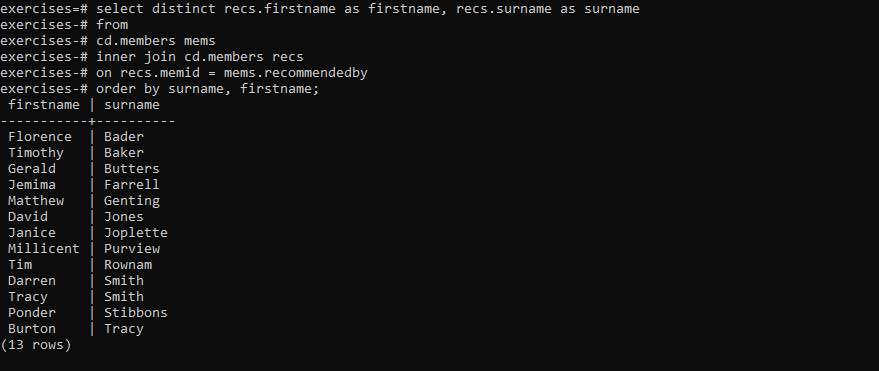
**13. How can you produce a list of the start times for bookings by members named 'David Farrell'?**



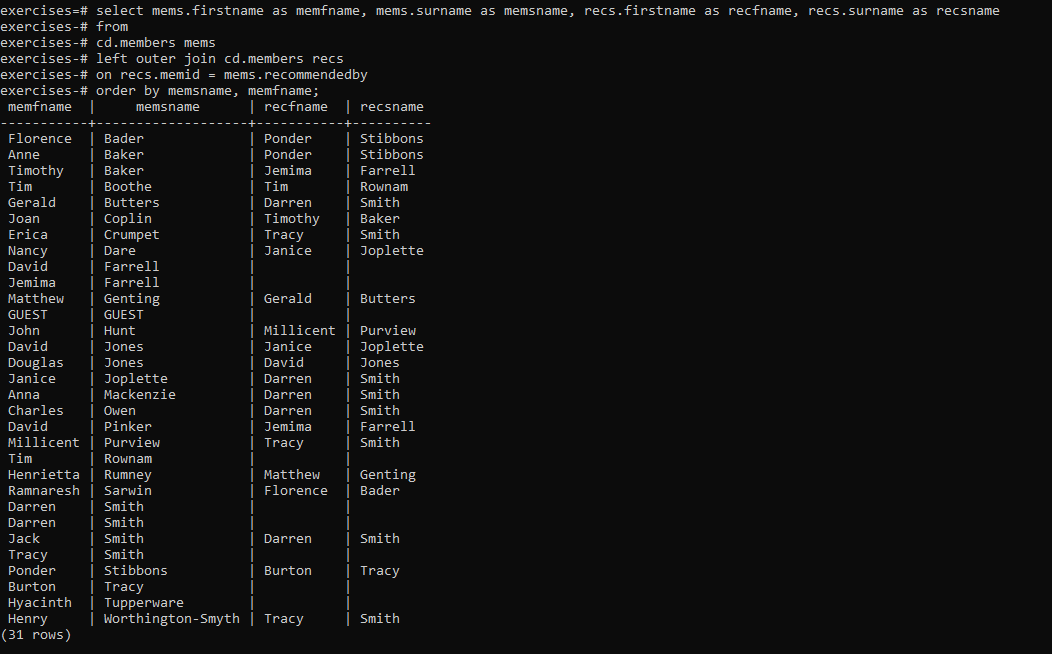
**14.** **How can you produce a list of the start times for bookings for tennis courts, for the date '2012-09-21'? Return a list of start time and facility name pairings, ordered by the time.**



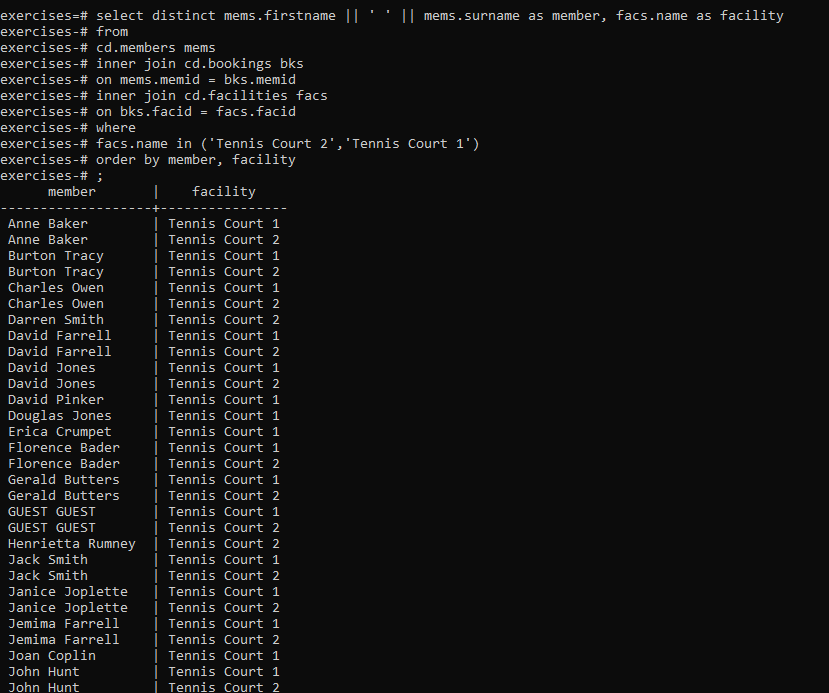
**15. How can you output a list of all members who have recommended another member? Ensure that there are no duplicates in the list, and that results are ordered by (surname, firstname).**



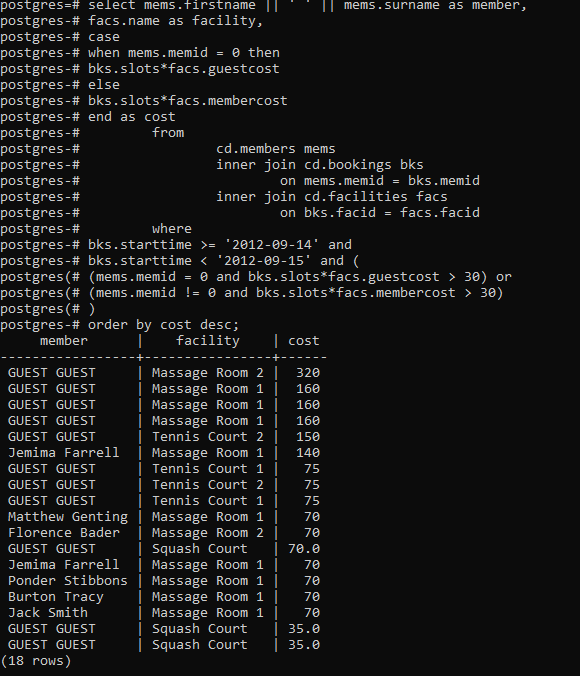
**16.** **How can you output a list of all members, including the individual who recommended them (if any)? Ensure that results are ordered by (surname, firstname).**



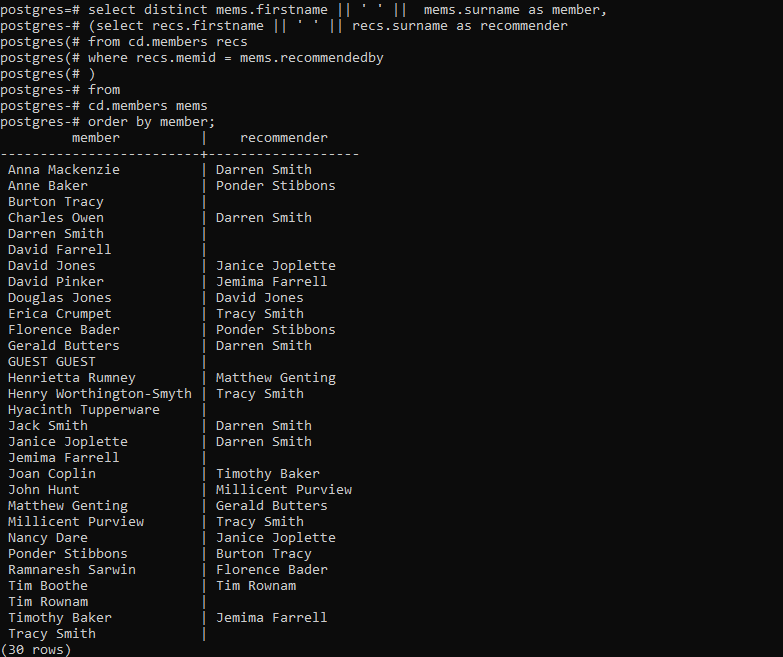
**17. How can you produce a list of all members who have used a tennis court? Include in your output the name of the court, and the name of the member formatted as a single column. Ensure no duplicate data, and order by the member name followed by the facility name.**



**19. How can you produce a list of bookings on the day of 2012-09-14 which will cost the member (or guest) more than $30? Remember that guests have different costs to members (the listed costs are per half-hour 'slot'), and the guest user is always ID 0. Include in your output the name of the facility, the name of the member formatted as a single column, and the cost. Order by descending cost, and do not use any subqueries.**

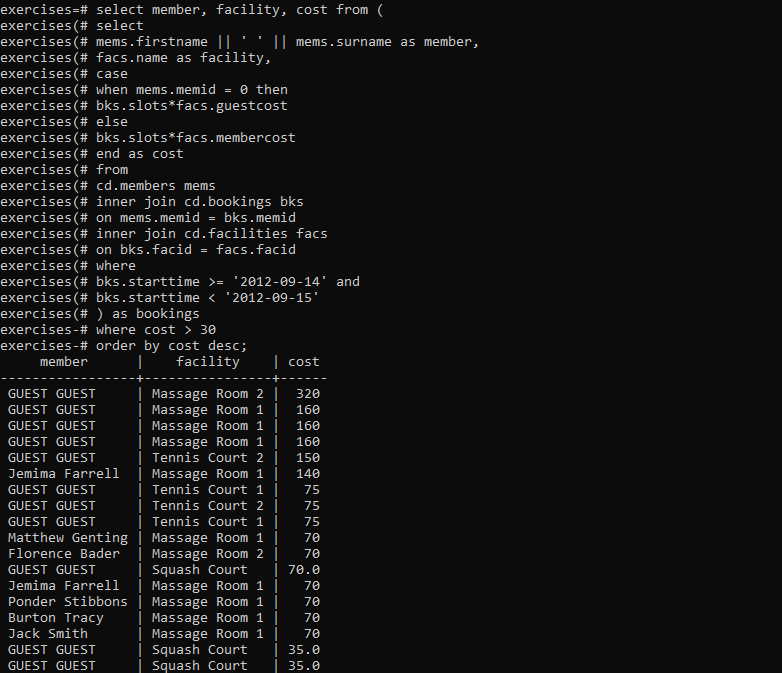


**20. How can you output a list of all members, including the individual who recommended them (if any), without using any joins? Ensure that there are no duplicates in the list, and that each firstname + surname pairing is formatted as a column and ordered.**



**21. The**[**Produce a list of costly bookings**](https://pgexercises.com/questions/joins/threejoin2.html)**exercise contained some messy logic: we had to calculate the booking cost in both the WHERE clause and the CASE statement. Try to simplify this calculation using subqueries. For reference, the question was:**

***How can you produce a list of bookings on the day of 2012-09-14 which will cost the member (or guest) more than $30? Remember that guests have different costs to members (the listed costs are per half-hour 'slot'), and the guest user is always ID 0. Include in your output the name of the facility, the name of the member formatted as a single column, and the cost. Order by descending cost.***



# //category-2:Modifying the data

**1. The club is adding a new facility - a spa. We need to add it into the facilities table. Use the following values:**

* **facid: 9, Name: 'Spa', membercost: 20, guestcost: 30, initialoutlay: 100000, monthlymaintenance: 800.**

**INSERT INTO FACILITIES(facid ,name,membercost,guestcost,initialoutlay,monthlymaintenance) VALUES (9,'Spa',20,30,100000,800);**



**2.** **In the previous exercise, you learned how to add a facility. Now you're going to add multiple facilities in one command. Use the following values:**

* **facid: 9, Name: 'Spa', membercost: 20, guestcost: 30, initialoutlay: 100000, monthlymaintenance: 800.**
* **facid: 10, Name: 'Squash Court 2', membercost: 3.5, guestcost: 17.5, initialoutlay: 5000, monthlymaintenance: 80.**

**Query:**

**insert into cd.facilities (facid,name,membercost,guestcost,initialoutlay,monthlymaintenance) values(9,'Spa',20,30,100000,800),(10,'Squash Court 2',3.5,17.5,5000,80);**

**3. Let's try adding the spa to the facilities table again. This time, though, we want to automatically generate the value for the next facid, rather than specifying it as a constant. Use the following values for everything else:**

* **Name: 'Spa', membercost: 20, guestcost: 30, initialoutlay: 100000, monthlymaintenance: 800.**

**Query:**

**insert into cd.facilities**

**(facid, name, membercost, guestcost, initialoutlay, monthlymaintenance)**

**select (select max(facid) from cd.facilities)+1, 'Spa', 20, 30, 100000, 800;**

**4. We made a mistake when entering the data for the second tennis court. The initial outlay was 10000 rather than 8000: you need to alter the data to fix the error.**

**Query:**

**update cd.facilities set initialoutlay=10000 where facid=1;**



**5. We want to increase the price of the tennis courts for both members and guests. Update the costs to be 6 for members, and 30 for guests.**

**Query:**

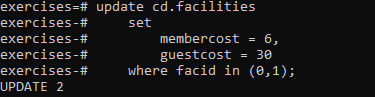
**update cd.facilities**

**set**

**membercost = 6,**

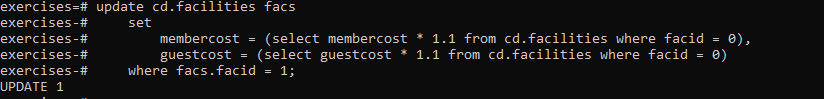
**guestcost = 30**

**where facid in (0,1);**



**6. We want to alter the price of the second tennis court so that it costs 10% more than the first one. Try to do this without using constant values for the prices, so that we can reuse the statement if we want to.**

**update cd.facilities facs set membercost = (select membercost \* 1.1 from cd.facilities where facid = 0), guestcost = (select guestcost \* 1.1 from cd.facilities where facid = 0) where facs.facid = 1;**



**7. As part of a clearout of our database, we want to delete all bookings from the cd.bookings table. How can we accomplish this?**

**Query:**

**Delete from cp.bookings;**



**8. We want to remove member 37, who has never made a booking, from our database. How can we achieve that?**

**Query:**

**delete from cd.members where memid=37;**



**9. In our previous exercises, we deleted a specific member who had never made a booking. How can we make that more general, to delete all members who have never made a booking?**

**Query:**

**delete from cd.members where memid not in (**

**select distinct memid from cd.bookings);**

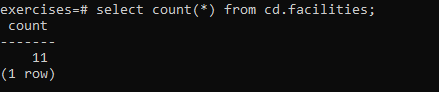


# //category-4:Aggregates

1. **For our first foray into aggregates, we're going to stick to something simple. We want to know how many facilities exist - simply produce a total count.**

**Query:**

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

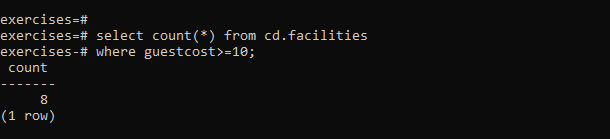


1. **Produce a count of the number of facilities that have a cost to guests of 10 or more.**

**Query:**

**select count(\*) from cd.facilities**

**where guestcost>=10;**



**3.Produce a count of the number of recommendations each member has made. Order by member ID.**

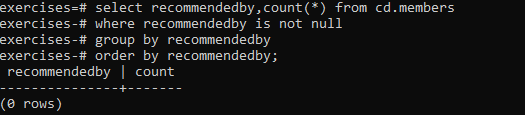
**Query:**

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

**where recommendedby is not null**

**group by recommendedby**

**order by recommendedby;**



**4.Produce a list of the total number of slots booked per facility. For now, just produce an output table consisting of facility id and slots, sorted by facility id.**

**Query:**

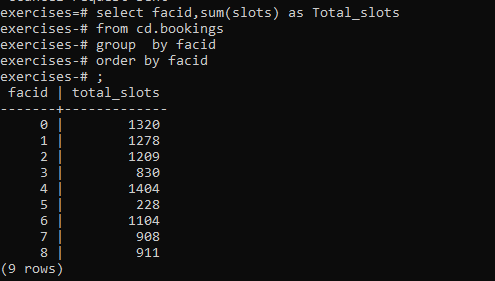
**select facid,sum(slots) as Total\_slots**

**from cd.bookings**

**group by facid**

**order by facid**

**;**



**5.Produce a list of the total number of slots booked per facility in the month of September 2012. Produce an output table consisting of facility id and slots, sorted by the number of slots.**

**Query:**

**select facid,sum(slots) as Total\_slots**

**from cd.bookings**

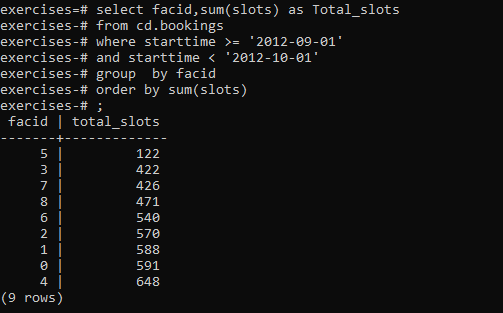
**where starttime >= '2012-09-01'**

**and starttime < '2012-10-01'**

**group by facid**

**order by sum(slots)**

**;**



**6.Produce a list of the total number of slots booked per facility per month in the year of 2012. Produce an output table consisting of facility id and slots, sorted by the id and month.**

**Query:**

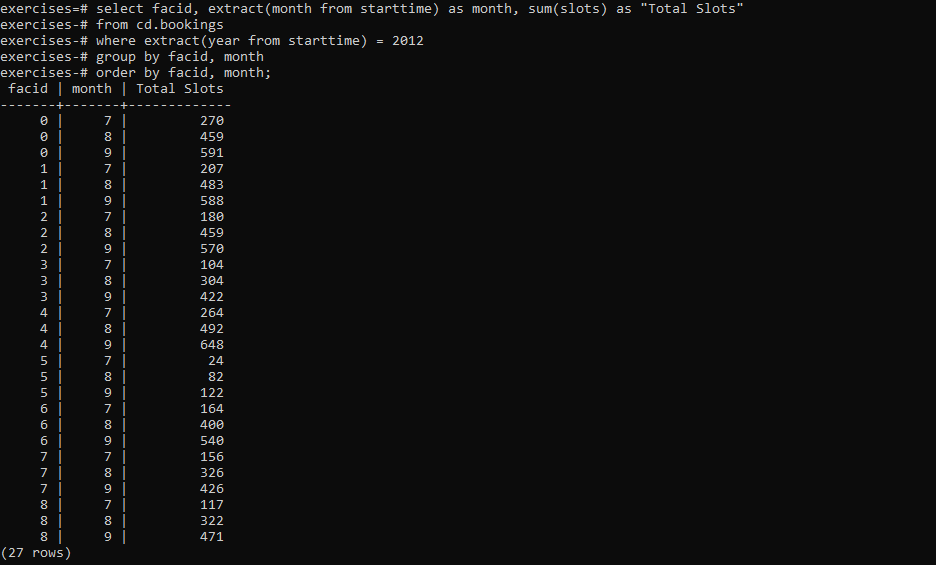
**select facid, extract(month from starttime) as month, sum(slots) as "Total Slots"**

**from cd.bookings**

**where extract(year from starttime) = 2012**

**group by facid, month**

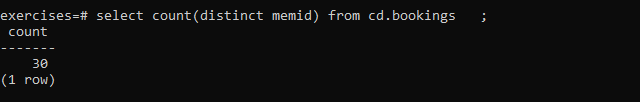
**order by facid, month;**



**7.Find the total number of members (including guests) who have made at least one booking.**

**Query:**

**select count(distinct memid) from cd.bookings ;**



**8.Produce a list of facilities with more than 1000 slots booked. Produce an output table consisting of facility id and slots, sorted by facility id.**

**Query:**

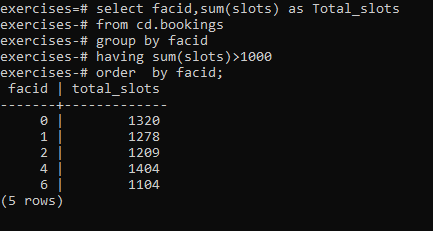
**select facid,sum(slots) as Total\_slots**

**from cd.bookings**

**group by facid**

**having sum(slots)>1000**

**order by facid;**



**9.Produce a list of facilities along with their total revenue. The output table should consist of facility name and revenue, sorted by revenue. Remember that there's a different cost for guests and members!**

**Query:**

**select facs.name, sum(slots \* case**

**when memid = 0 then facs.guestcost**

**else facs.membercost**

**end) as revenue**

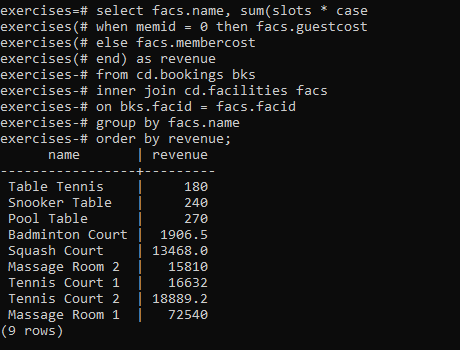
**from cd.bookings bks**

**inner join cd.facilities facs**

**on bks.facid = facs.facid**

**group by facs.name**

**order by revenue;**



**10.Produce a list of facilities with a total revenue less than 1000. Produce an output table consisting of facility name and revenue, sorted by revenue. Remember that there's a different cost for guests and members!**

**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**

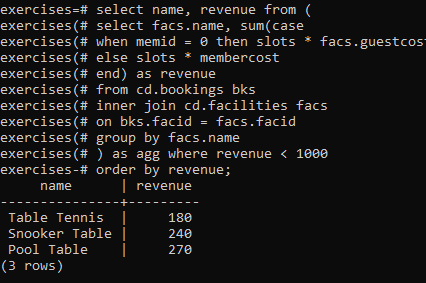
**inner join cd.facilities facs**

**on bks.facid = facs.facid**

**group by facs.name**

**) as agg where revenue < 1000**

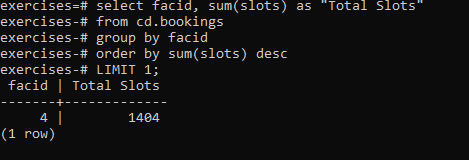
**order by revenue;**



**11.Output the facility id that has the highest number of slots booked. For bonus points, try a version without a LIMIT clause. This version will probably look messy!**

**Query:**

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



**12.Produce a list of the total number of slots booked per facility per month in the year of 2012. In this version, include output rows containing totals for all months per facility, and a total for all months for all facilities. The output table should consist of facility id, month and slots, sorted by the id and month. When calculating the aggregated values for all months and all facids, return null values in the month and facid columns.**

**Query:**

**select facid, extract(month from starttime) as month, sum(slots) as slots**

**from cd.bookings**

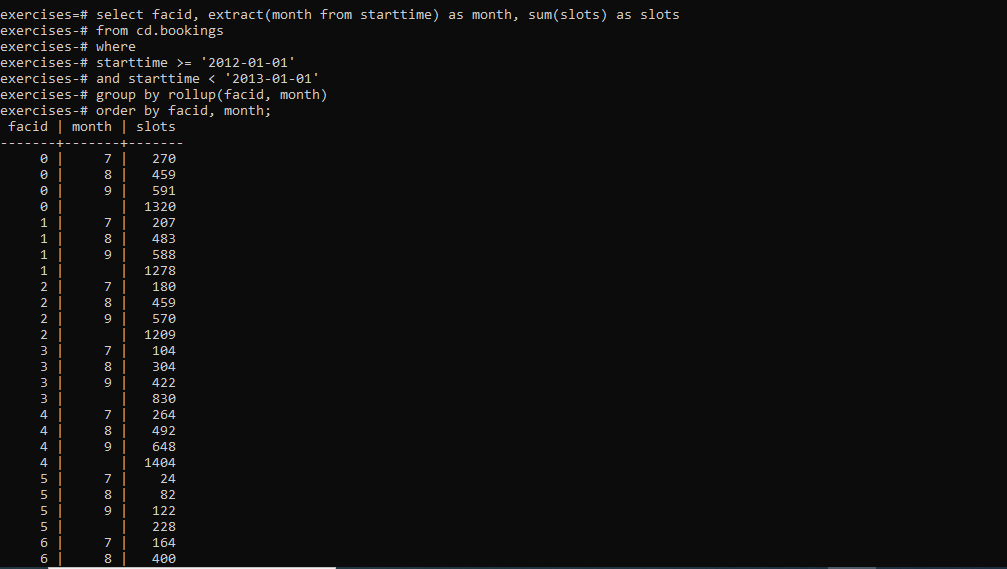
**where**

**starttime >= '2012-01-01'**

**and starttime < '2013-01-01'**

**group by rollup(facid, month)**

**order by facid, month;**



**13.Produce a list of the total number of *hours* booked per facility, remembering that a slot lasts half an hour. The output table should consist of the facility id, name, and hours booked, sorted by facility id. Try formatting the hours to two decimal places.**

**Query:**

**select facs.facid, facs.name,**

**trim(to\_char(sum(bks.slots)/2.0, '9999999999999999D99')) as "Total Hours"**

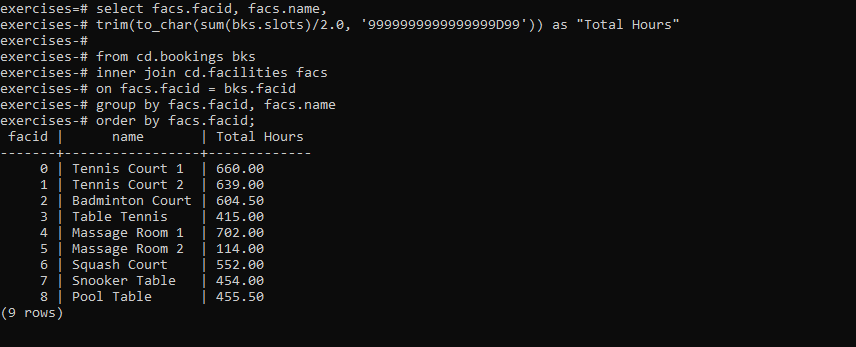
**from cd.bookings bks**

**inner join cd.facilities facs**

**on facs.facid = bks.facid**

**group by facs.facid, facs.name**

**order by facs.facid;**



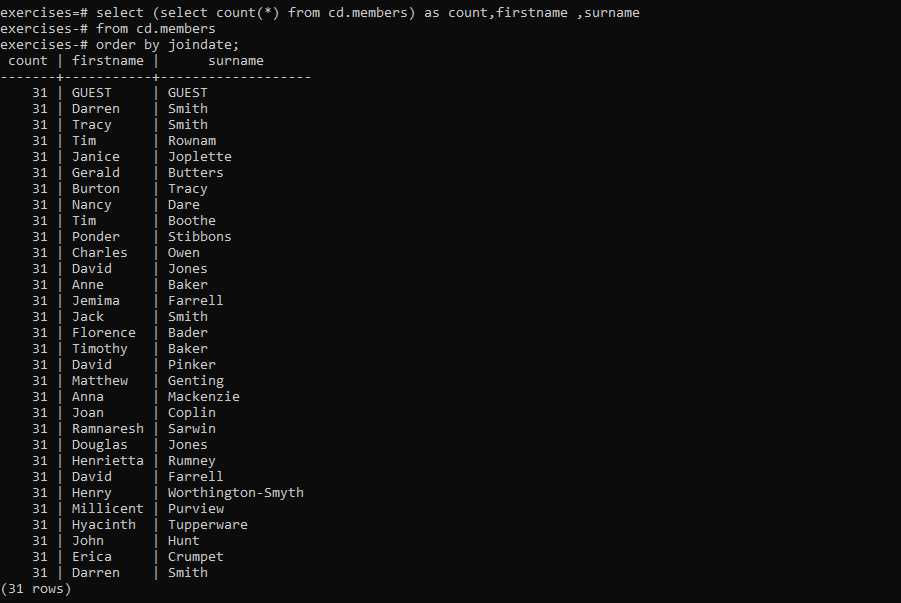
**14. Produce a list of member names, with each row containing the total member count. Order by join date, and include guest members.**

**Order:**

**select (select count(\*) from cd.members) as count,firstname ,surname**

**from cd.members**

**order by joindate;**



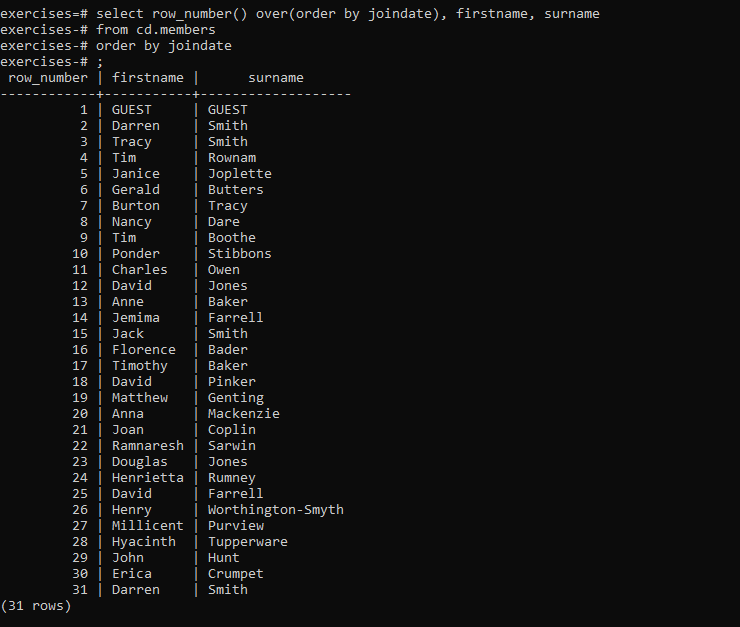
**15.Produce a monotonically increasing numbered list of members (including guests), ordered by their date of joining. Remember that member IDs are not guaranteed to be sequential.**

**Query:**

**select row\_number() over(order by joindate), firstname, surname**

**from cd.members**

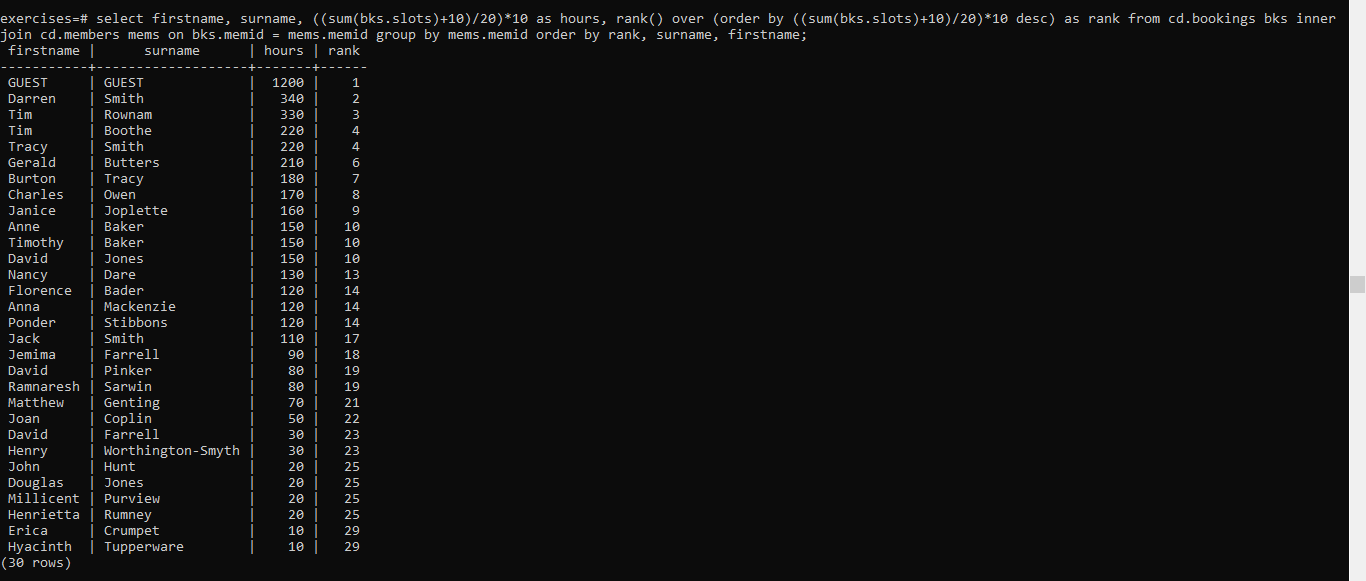
**order by joindate ;**



**17. Produce a list of members (including guests), along with the number of hours they've booked in facilities, rounded to the nearest ten hours. Rank them by this rounded figure, producing output of first name, surname, rounded hours, rank. Sort by rank, surname, and first name.**

**Query:**

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



**19.Classify facilities into equally sized groups of high, average, and low based on their revenue. Order by classification and facility name.**

**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**

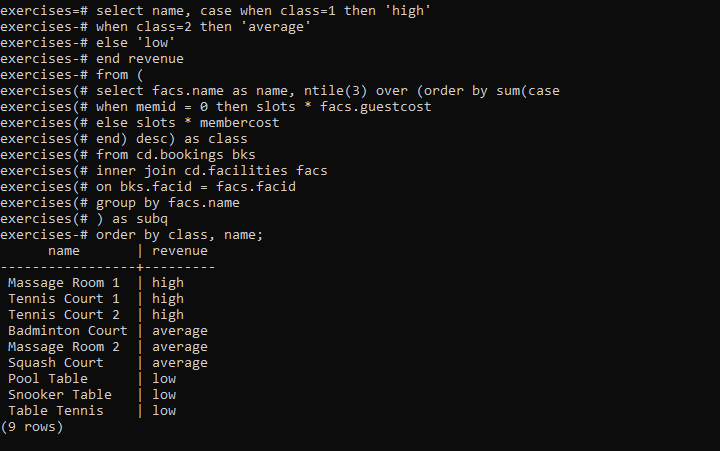
**inner join cd.facilities facs**

**on bks.facid = facs.facid**

**group by facs.name**

**) as subq**

**order by class, name;**

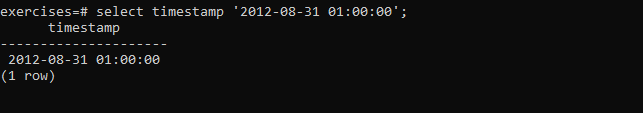


# //Category-5:Date

1. **Produce a timestamp for 1 a.m. on the 31st of August 2012.**

Query:

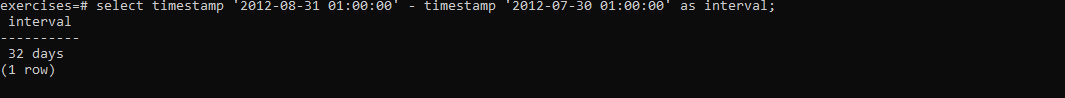
**select timestamp '2012-08-31 01:00:00';**



1. **Find the result of subtracting the timestamp '2012-07-30 01:00:00' from the timestamp '2012-08-31 01:00:00'**

**Query:**

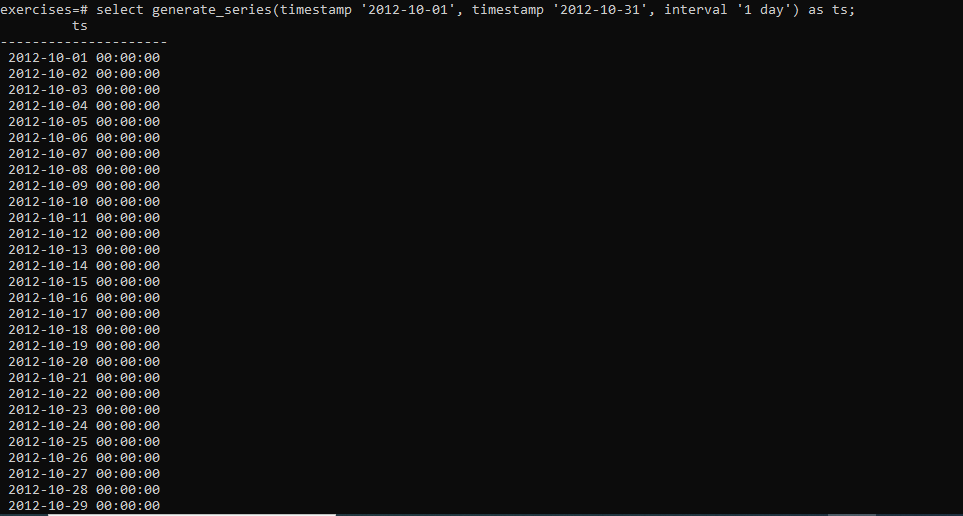
**select timestamp '2012-08-31 01:00:00' - timestamp '2012-07-30 01:00:00' as interval;**



1. Produce a list of all the dates in October 2012. They can be output as a timestamp (with time set to midnight) or a date.

**Query:**

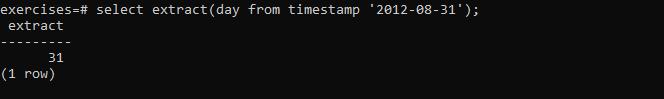
**select** generate\_series(timestamp '2012-10-01', timestamp '2012-10-31', interval '1 day') **as** ts;



1. Get the day of the month from the timestamp '2012-08-31' as an integer.

**Query:**

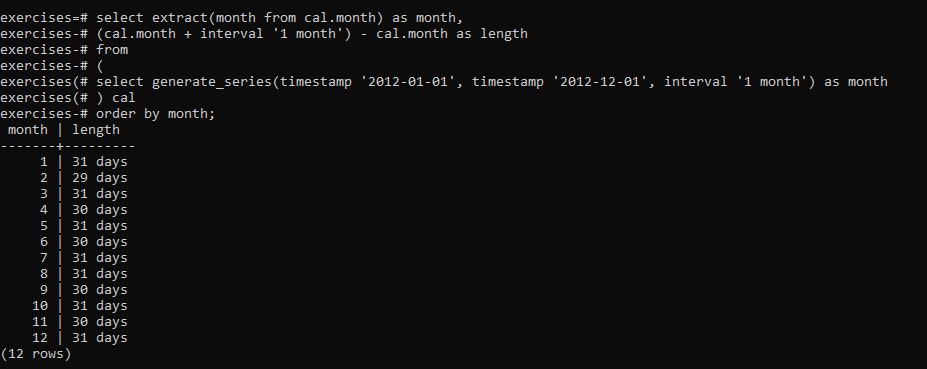
**select extract(day from timestamp '2012-08-31');**



**6.For each month of the year in 2012, output the number of days in that month. Format the output as an integer column containing the month of the year, and a second column containing an interval data type.**

**Query:**

**select extract(month from cal.month) as month, (cal.month + interval '1 month') - cal.month as length from ( select generate\_series(timestamp '2012-01-01', timestamp '2012-12-01', interval '1 month') as month ) cal order by month;**

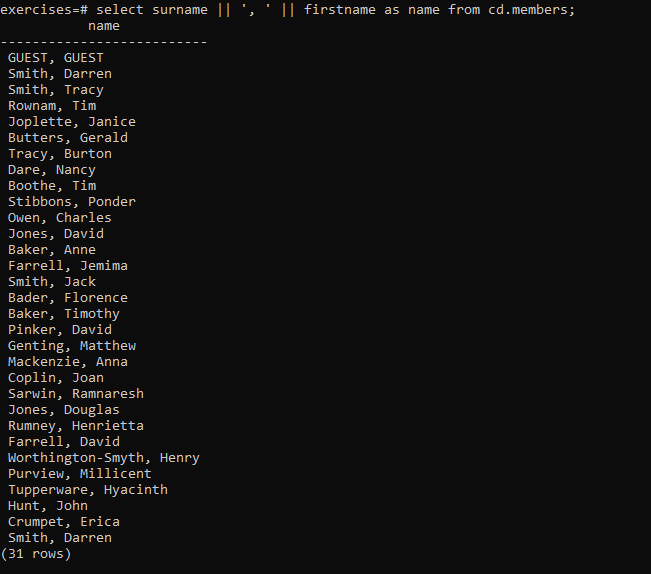


# //Cattegory:String

**1.Output the names of all members, formatted as 'Surname, Firstname'**

**Query:**

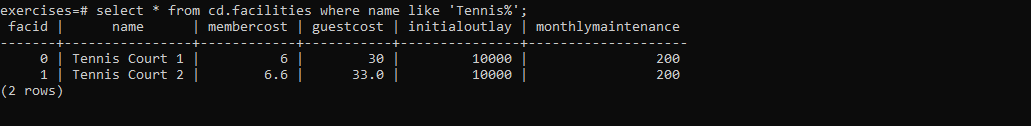
select surname || ', ' || firstname as name from cd.members;



1. Find all facilities whose name begins with 'Tennis'. Retrieve all columns.

Query:

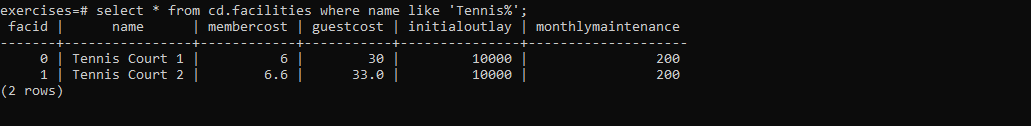
s**elect** \* **from** cd.facilities **where** name **like** 'Tennis%';



1. Perform a case-insensitive search to find all facilities whose name begins with 'tennis'. Retrieve all columns.

Query:

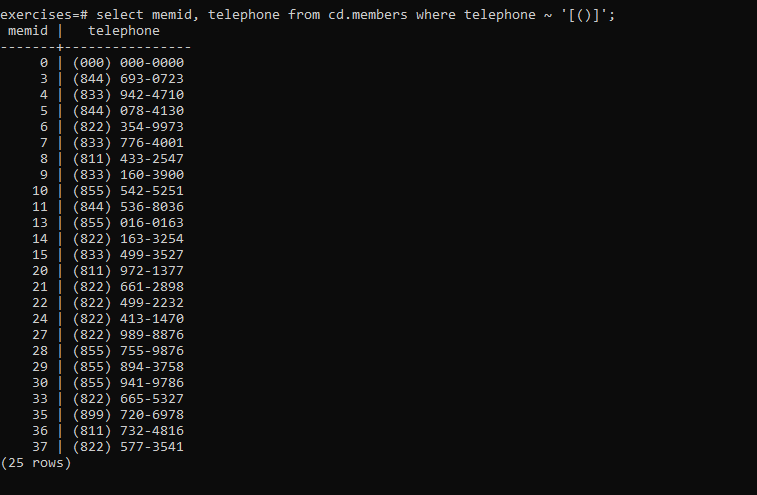
**select** \* **from** cd.facilities **where** upper(name) **like** 'TENNIS%';



1. You've noticed that the club's member table has telephone numbers with very inconsistent formatting. You'd like to find all the telephone numbers that contain parentheses, returning the member ID and telephone number sorted by member ID.

Query:

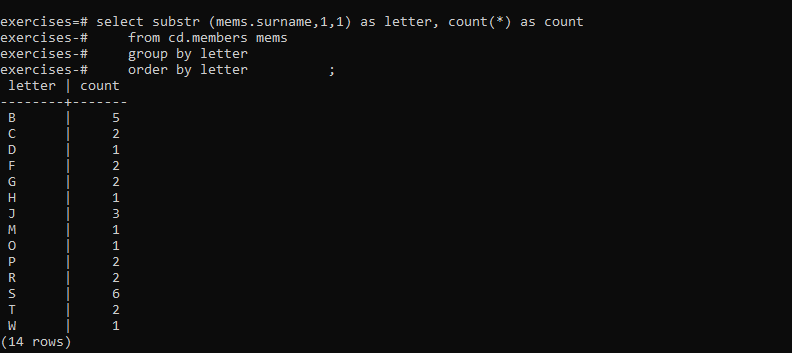
**select** memid, telephone **from** cd.members **where** telephone ~ '[()]';



1. You'd like to produce a count of how many members you have whose surname starts with each letter of the alphabet. Sort by the letter, and don't worry about printing out a letter if the count is 0.

Query:

**select** substr (mems.surname,1,1) **as** letter, count(\*) **as** count **from** cd.members mems **group** **by** letter **order** **by** letter;



1. The telephone numbers in the database are very inconsistently formatted. You'd like to print a list of member ids and numbers that have had '-','(',')', and ' ' characters removed. Order by member id.

Query:

**select** memid, translate(telephone, '-() ', '') **as** telephone **from** cd.members **order** **by** memid;

