1. Listing of children and their associated carer

create view Childlist as

select

concat\_ws(' ', child\_fname, child\_sname) as 'Child Name',

concat\_ws(' ', carer\_fname, carer\_sname) as 'Carer Name',

carer\_phone as 'Carer Phone'

from Child, Carer

where child\_carer = carer\_id

order by child\_sname, child\_fname

I first run the query without a WHERE clause and it returned multiple entries for every child record. Having also used the CREATE VIEW statement in the query, I generated an incorrect view in MySQL Workbench. I had to delete the view and re-run the query, ensuring that it retrieved unique instances of every child record by matching the carer\_id values in the Carer table with the child\_carer values in the Child table.

In order to manually check that the number of children returned for my query (eighteen) matched the number of unique instances of the Child entity (also eighteen), I executed the following: *select \* Child*. To do the same for the Carer entity, I decided to group the results by carer\_id and found eleven rows. This matched the number of carers with associated children: *select \* from Carer, Child where carer\_id = child\_carer*.

The Child Name and Carer Name columns could also be created with the CONCAT function and a separator ‘ ‘ in between the column names instead of the CONCAT\_WS function. Though, the latter would have been even more efficient if there were more columns to concatenate.

1. **Count of children each carer is responsible for**

select

concat\_ws(' ', carer\_fname, carer\_sname) as 'Carer Name',

count(child\_id) as 'No. Children'

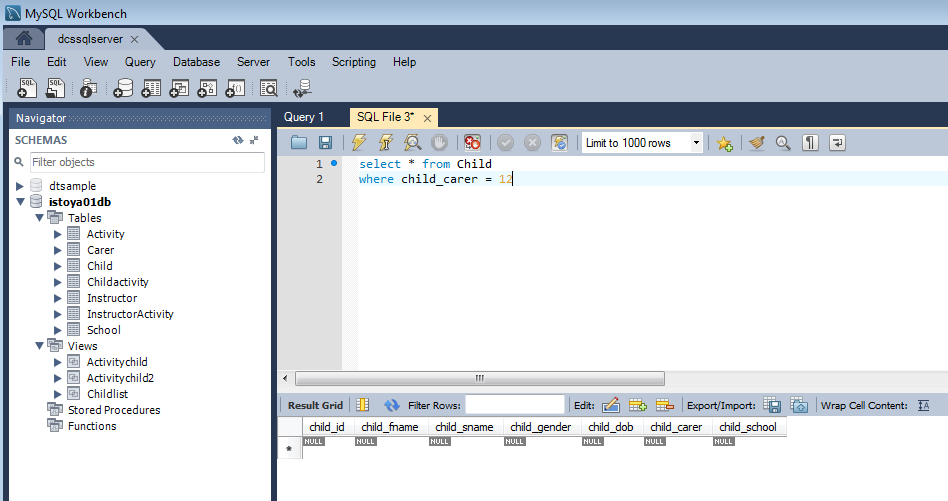
from Carer

left join Child on carer\_id = child\_carer

group by carer\_id

The Carer table was the parent table in this instance soI grouped the query results by carer\_id. I also needed to include carers with no associated children. I tried a RIGHT JOIN that did the opposite but I managed to retrieve all rows in the Carer (left) table with the matched rows in the Child (right) with a LEFT JOIN.

I started counting all records in the Child table, however the carer with no associated children appeared in the results as though a child was linked to their record. Therefore, I had to count only the child\_id values. In order to manually check that my results were correct, I had to find out what the carer\_id was for the carer with no associated children: *select \* from Carer*. Using the carer\_id I searched for any associated children: *select \* from Child where child\_carer = 12*. No results were found:



1. **A register for a specified activity**

set @activityname = 'Art';

/\* Assign a different value to the @activityname variable and re-run the query in order to compile a register for every activity e.g. ‘Football’, ‘IT’ etc. \*/

set @activitynum =

(select activity\_id

from Activity

where activity\_name = @activityname);

select @activityname as 'Activity Name',

concat\_ws(' ', child\_fname, child\_sname) as 'Child Name',

concat\_ws(' ', carer\_fname, carer\_sname) as 'Carer Name',

carer\_phone as 'Carer Phone'

from Activity A

join Childactivity CA on CA.activity\_id = A.activity\_id

join Child CH on CH.child\_id = CA.child\_id

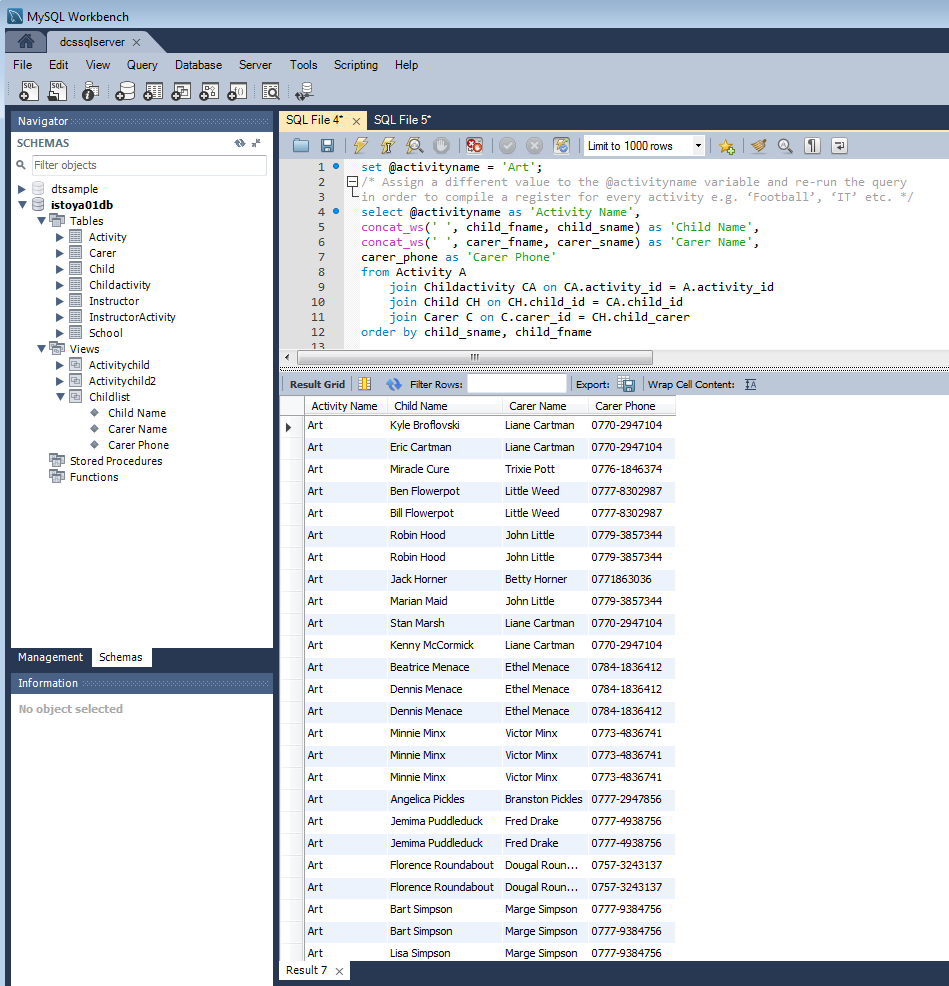
join Carer C on C.carer\_id = CH.child\_carer

where CA.activity\_id = @activitynum

order by child\_sname, child\_fname

The @activityname variable is the named location in the memory of the Playscheme database for the activity on which I am running the query. I can retrieve results for a different activity with the same criteria if I update the @activityname value. I set a second variable, @activitynum, to store the unique identifier (activity\_id) for the specified activity (Art). I have used the resulting value in the WHERE clause where I match it to the activity\_id attribute of the Childactivity table. I received a few errors due to ambiguity with the column names so they had to be qualified. I assigned an alias for every joined.

Initially, I designed the query without an @activitynum variable but I had to run an additional query to confirm the activity\_id. Without an @activitynum value in the WHERE clause the query would have also returned duplicate rows:



The results can be checked as follows:

select \* from Activity

* activity\_id for Art is 1

select \* from Childactivity where activity\_id = 1

* the first row reads child\_id = 1

select \* from Child where child\_id = 1

* the first row reads Jack Horner with child\_carer = 1

select \* from Carer where carer\_id = 1

* the first row reads Betty Horner

1. **Total amount payable by each carer**

select

concat\_ws(' ', carer\_title, carer\_fname, carer\_sname) as 'Carer',

count(A.activity\_id) as 'Activities Taken',

round(sum(activity\_fee),2) as 'Total Due'

from Carer C, Child CH, Childactivity CA, Activity A

where C.carer\_id = CH.child\_carer

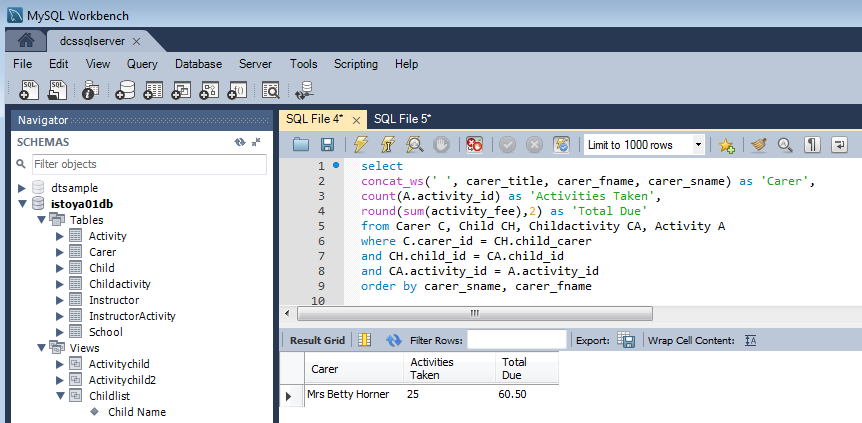
and CH.child\_id = CA.child\_id

and CA.activity\_id = A.activity\_id

group by C.carer\_id

order by carer\_sname, carer\_fname

I forgot to group the results by carer\_id from the Carer table at first so the query returned only the first carer in the entity as per the screenshot below.



I did not need to include carers without associated children, and instead of using a JOIN clause, I added this WHERE clause: carer\_id = child\_carer. In order to avoid duplication, I only looked for instances where the activity\_id values in the Childactivity table matched the activity\_id values in the Activity table. I retrieved a list of the children and their activity by matching the child\_id in the Childactivity table with the child\_id in the Child table.