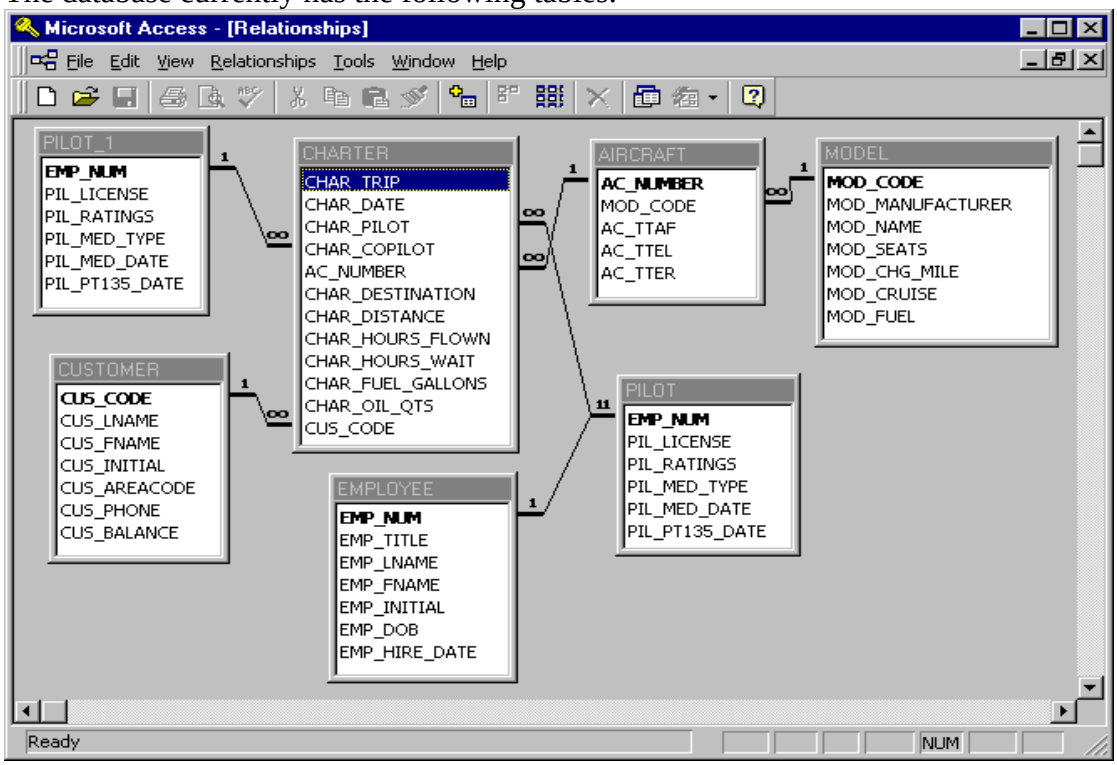
Data Exploration
The Robcor Case Study

ROBCOR, Inc. provides "on demand" aviation charters, using a mix of different airplane and airplane types. Because ROBCOR, Inc., has grown rapidly, it has hired you to be its first database manager. Your first and critical assignment is to develop a decision support system to analyze the charter data. The charter operations manager wants to be able to analyze charter data such as total hours flown, total fuel used, and total revenue (charter distance x model charge per mile). She would also like to be able to drill-down by pilot, airplane model, and time periods. The main requirements for this database are to:

a. Show the total revenue each month/year
b. Show the total hours flown by each pilot
c. Show the total fuel used by each airplane model.

The database currently has the following tables:



The tables can be copied from the `dw` account, using:

```
Select * from dw.<table_name>;
```

Or

```
Create Table <your_table_name> As
Select …
From dw.<table_name>
Where …
```

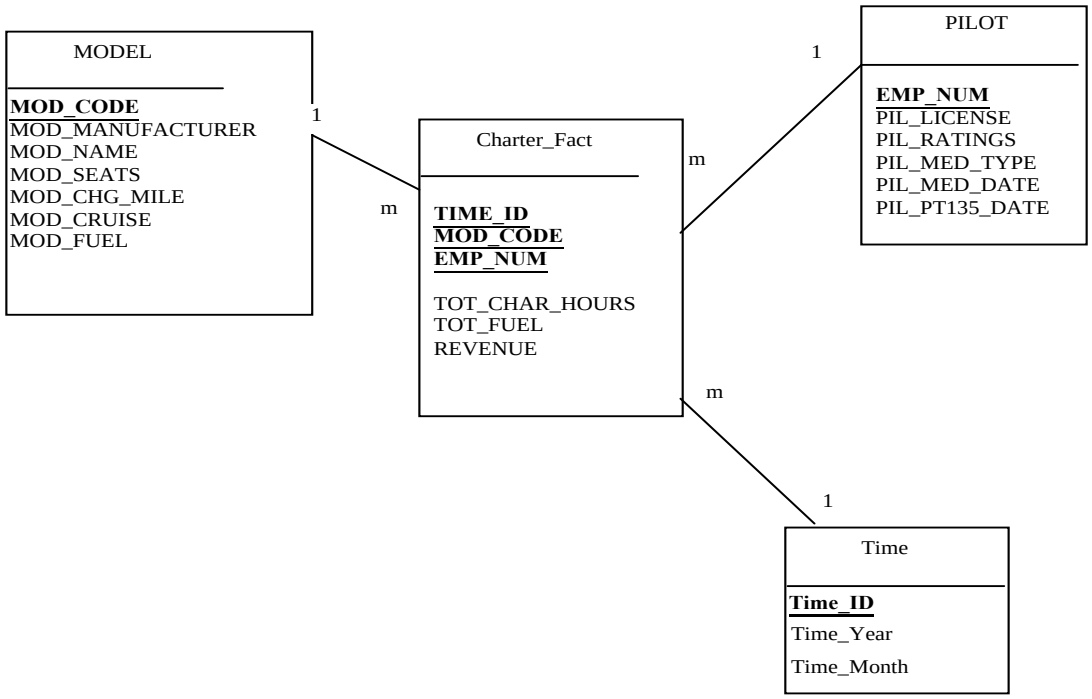
Given these requirements, complete the following:

1. Create a star schema for the charter data.
2. Define the dimensions and attributes for the charter operation's star schema.
3. Define the SQL statements for the implementation of the star schema.

4. Write the SQL statements to produce the following reports:
    a. Show the total revenue each year
    b. Show the total hours flown by each pilot
    c. Show the total fuel used by each aircraft model

Solutions:

The Star Schema



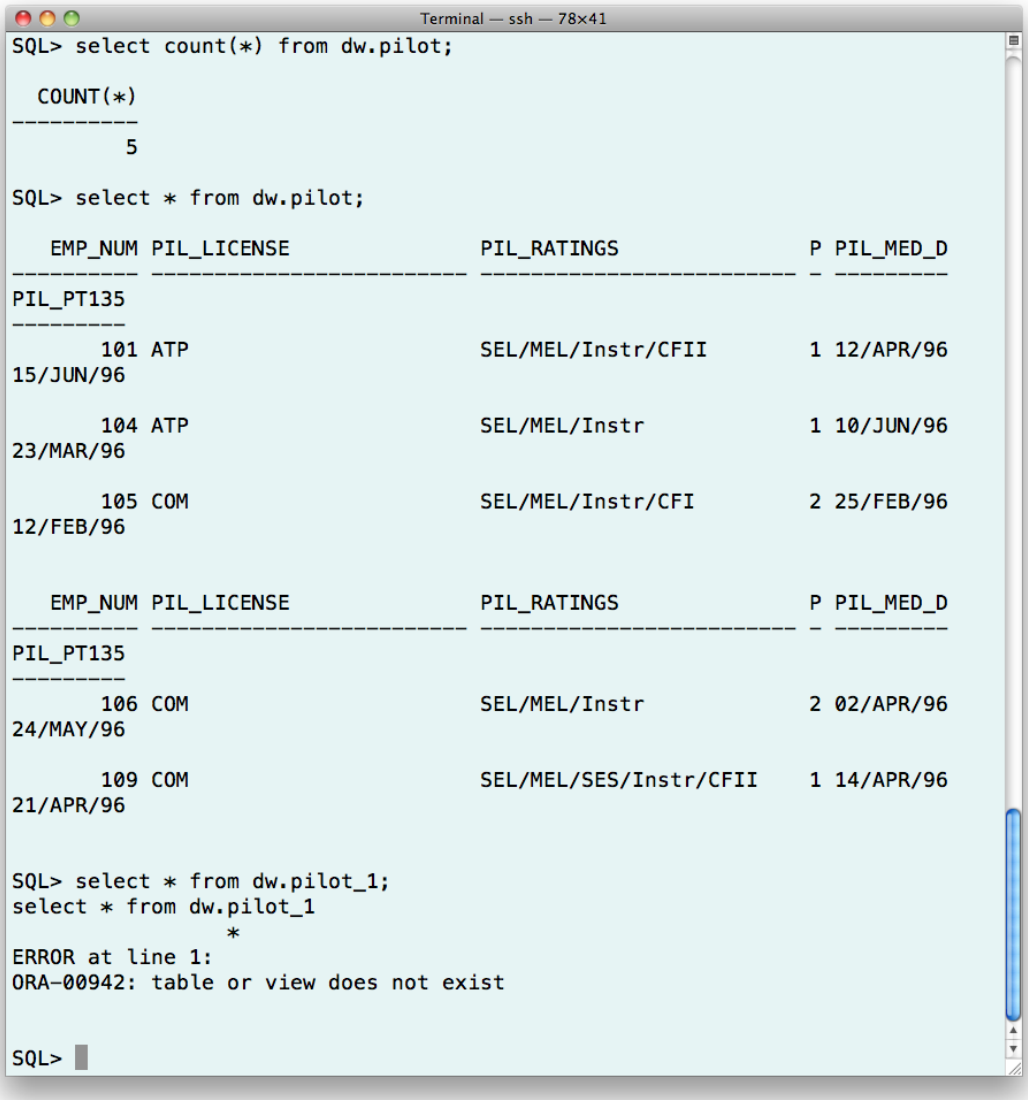

```
--First create the dimensions
create table time As
select Distinct to_char(char_date, 'YYYYMM') as Time_ID,
                to_char(char_date, 'Month') as Time_Month,
                to_char(char_date, 'YYYY') as Time_Year
from dw.Charter;

create table model as
select * from dw.model;

create table pilot as
select * from dw.pilot;

--Second, create the Charter_fact (the fact table) table
create table charter_fact as
select C.Char_Pilot as EMP_Num,
       M.Mod_Code,
       to_char(C.Char_Date, 'YYYYMM') as Time_ID,
       sum(C.Char_Hours_Flown) as Tot_Char_Hours,
       sum(C.Char_Fuel_Gallons) as Tot_Fuel,
       sum(C.Char_Distance * M.Mod_chg_mile) as Revenue
from   dw.Charter C, dw.Model M, dw.Aircraft A
where  C.AC_Number=A.AC_Number and A.Mod_Code=M.Mod_Code
group by C.Char_Pilot, M.Mod_Code, to_char(C.Char_Date, 'YYYYMM');
```

Now, we need to explore the data, both operational data, and data warehouse data. **Data exploration** is always necessary, because we will not be able to determine whether the results of our data warehouse are correct or not by just looking at the SQL above.

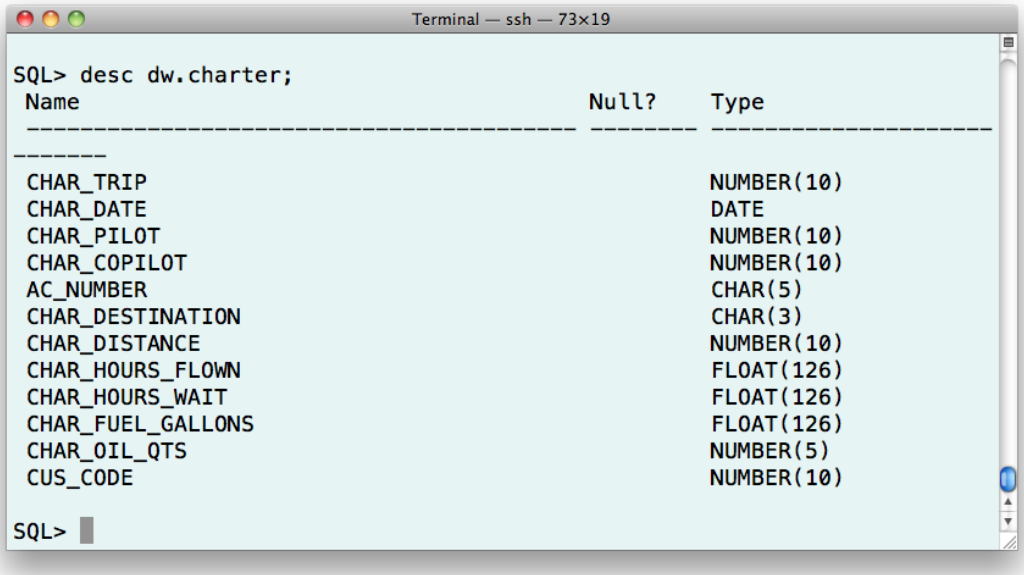
You first need to explore all tables in the operational database, including finding out what the operational data is about. Here is what I found:

1. Where is Pilot_1 table?

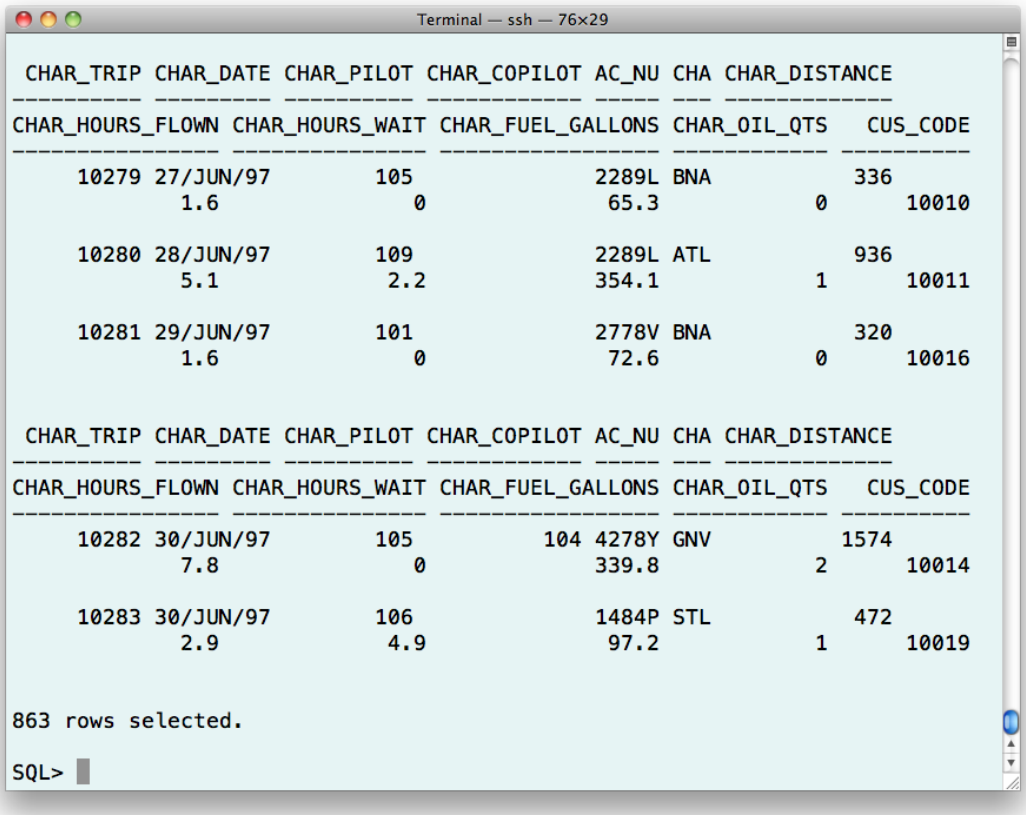
In the E/R diagram above, there is Pilot_1 entity. But I could not find the table. Pilot table is there, but not Pilot_1 table. There are five records in Pilot table.



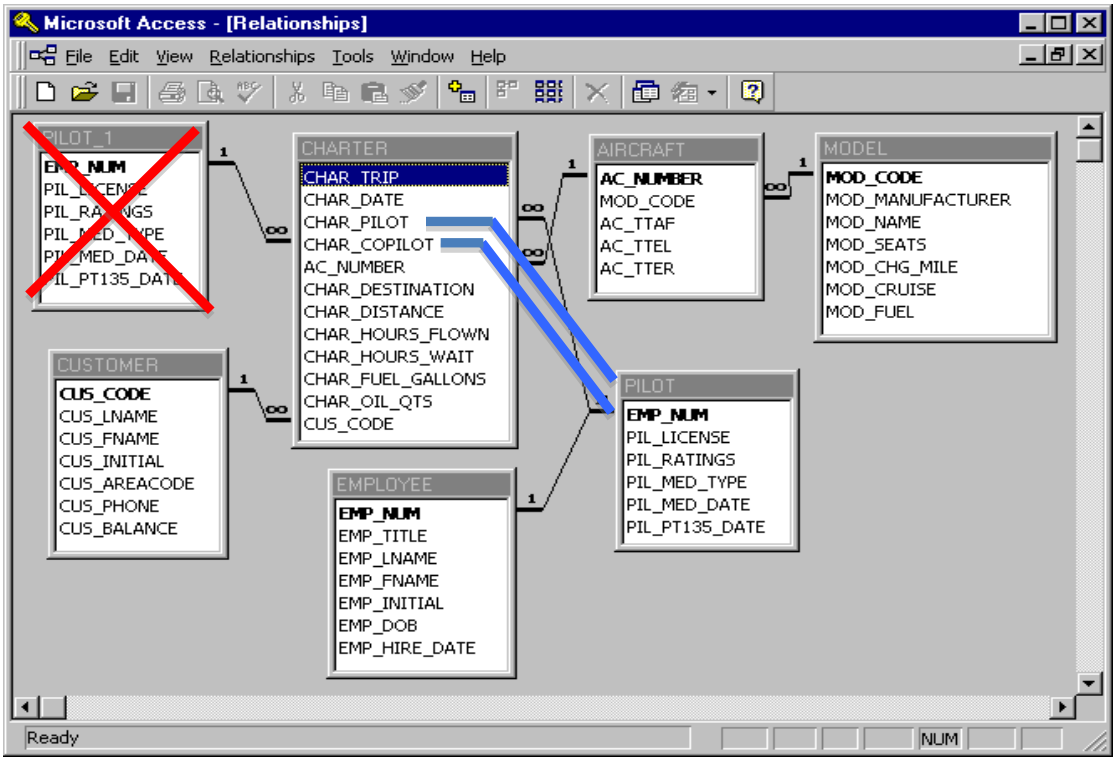
However, when I do `Select * From dw.Charter`, the attributes are all complete, including Char_pilot and Char_copilot attributes.



After checking dw.charter table (`select * from dw.charter`), this is what I've got:

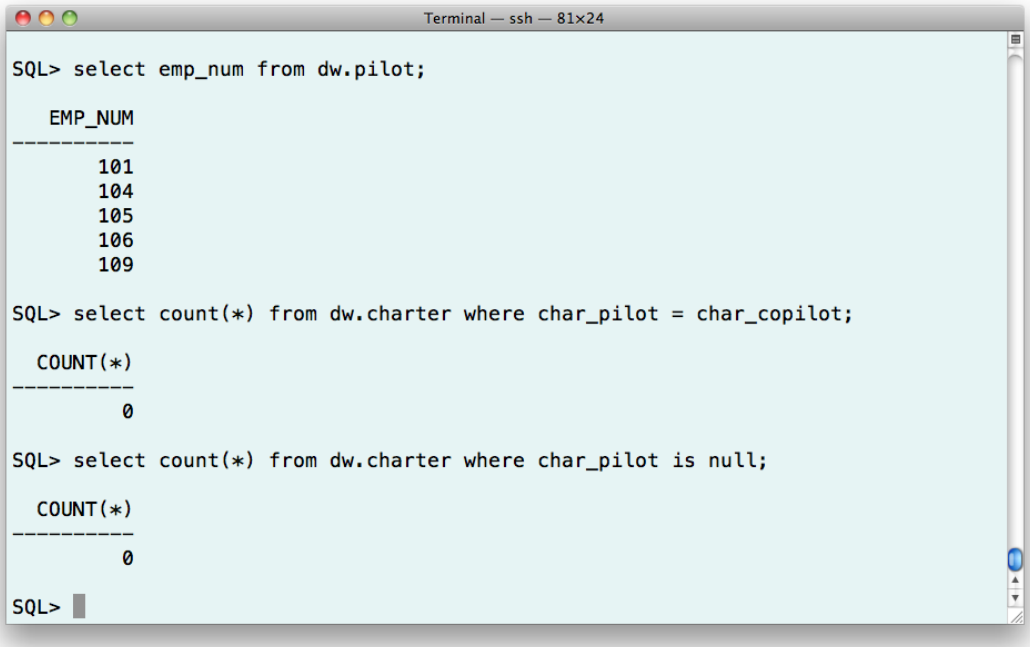


It seems that Char_pilot and Char_copilot are drawing from the same table, that is table Pilot. Some records in dw.charter have entries in the Char_copilot column. So, the conclusion that I can draw is that the give E/R diagram is not accurate. It is perhaps the way the designer drew the E/R diagram was like that, to indicate that table Pilot is used twice by the Charter entity. Hence, a revised E/R diagram could be drawn like this:

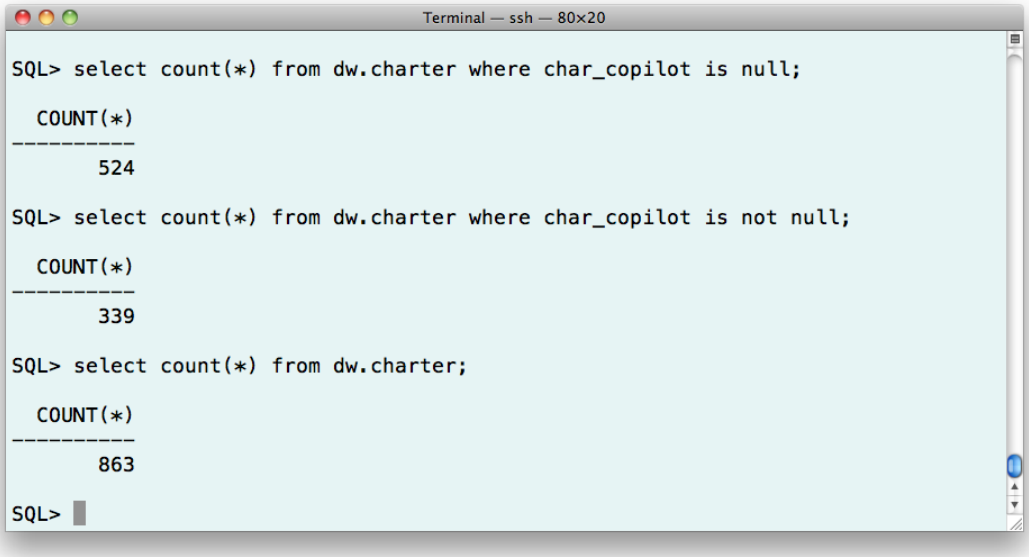


While on the subject of pilot and copilot, let's explore more:

The first one is to check whether there are records where the copilot is the pilot (i.e. the entry in char_copilot is the same as in char_pilot). I found nothing. So this is correct. The second check is to find out whether there are records without pilot. Also nothing. So this is also correct.



Now, let's check the copilots. We know that not all flights have copilots. Let's check how many flights with copilots, and how many flights without copilots, and see whether the sum of these match with the total number of records in dw.charter.

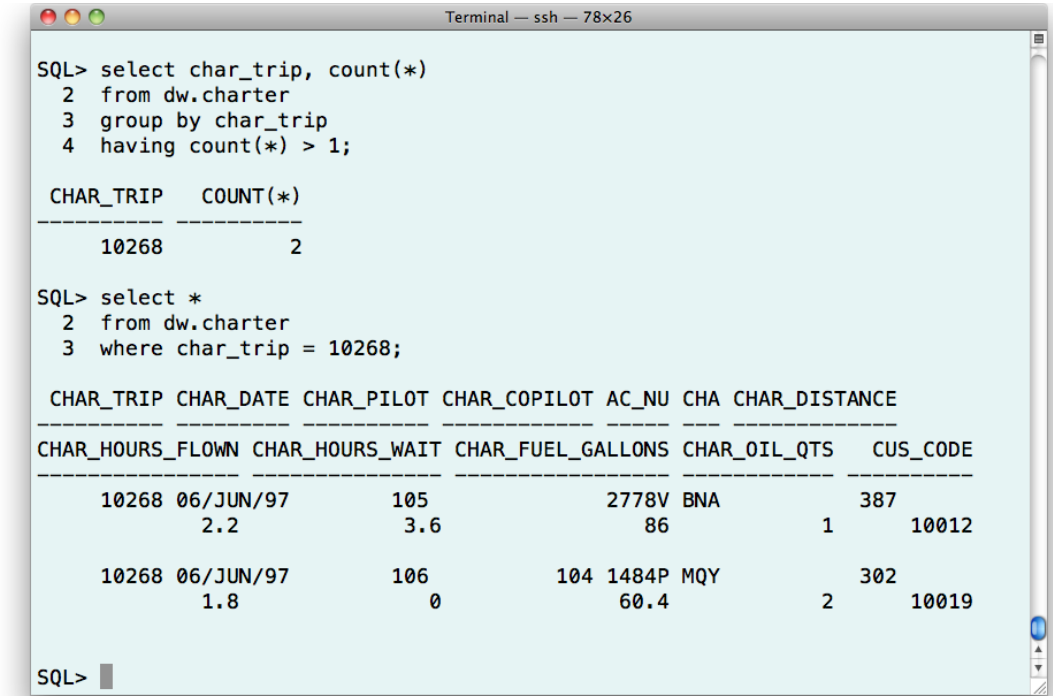


Everything about pilot and copilot seem to be fine. So, the only issue was with the E/R diagram.

The following shows number of charter flights for pilot 101 as a pilot, and as a copilot. The sum shows that, in this case, employee 101 cannot be pilot and a copilot for the same charter record. So, this is correct.

2. Duplicate Char_Trip

I found two different records in dw.charter sharing the same char_trip. By looking at these two records, it seems that these are two different charter records. Since in our data warehouse, char_trip does not affect the calculations in the fact table, we should not worry about this inconsistent data. So, the input data is not clean, but the unclean input data does not affect our data warehouse.



3. Should we consider copilot when calculating the fact table?

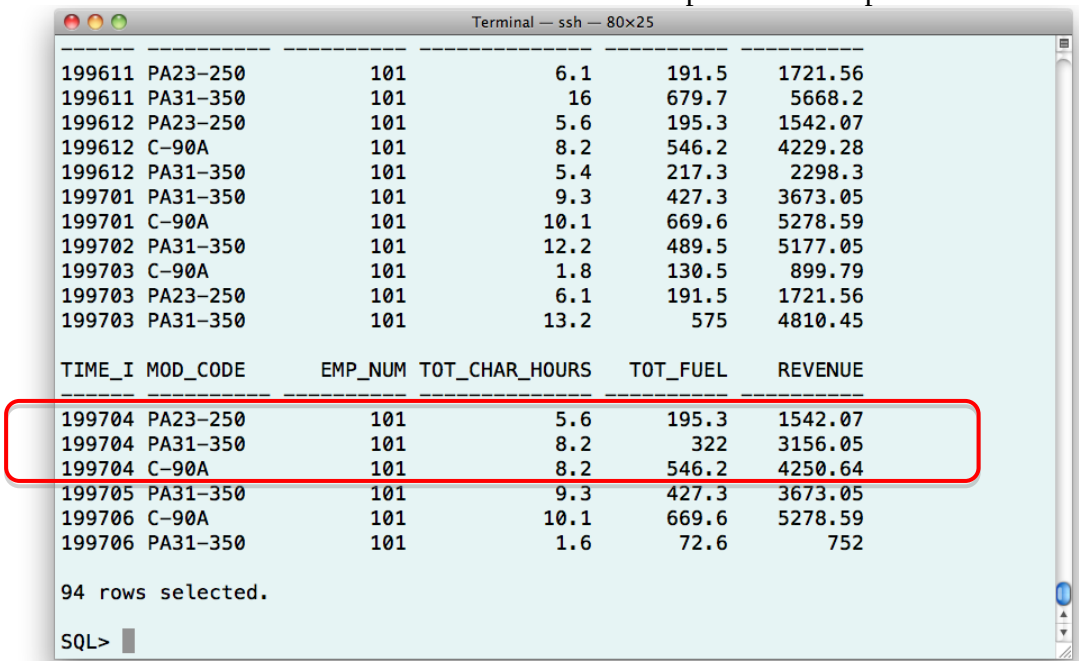
When Charter_fact is created, we do not consider copilot. The SQL to create Charter_fact is as follows:

```
create table charter_fact as
select C.Char_Pilot as EMP_Num,
       M.Mod_Code,
       to_char(C.Char_Date, 'YYYYMM') as Time_ID,
       sum(C.Char_Hours_Flown) as Tot_Char_Hours,
       sum(C.Char_Fuel_Gallons) as Tot_Fuel,
       sum(C.Char_Distance * M.Mod_chg_mile) as Revenue
from   dw.Charter C, dw.Model M, dw.Aircraft A
where  C.AC_Number=A.AC_Number and A.Mod_Code=M.Mod_Code
group by C.Char_Pilot, M.Mod_Code, to_char(C.Char_Date, 'YYYYMM');
```

Now let's explore pilot 101:

```
select *
from charter_fact
where emp_num=101
order by time_id;
```

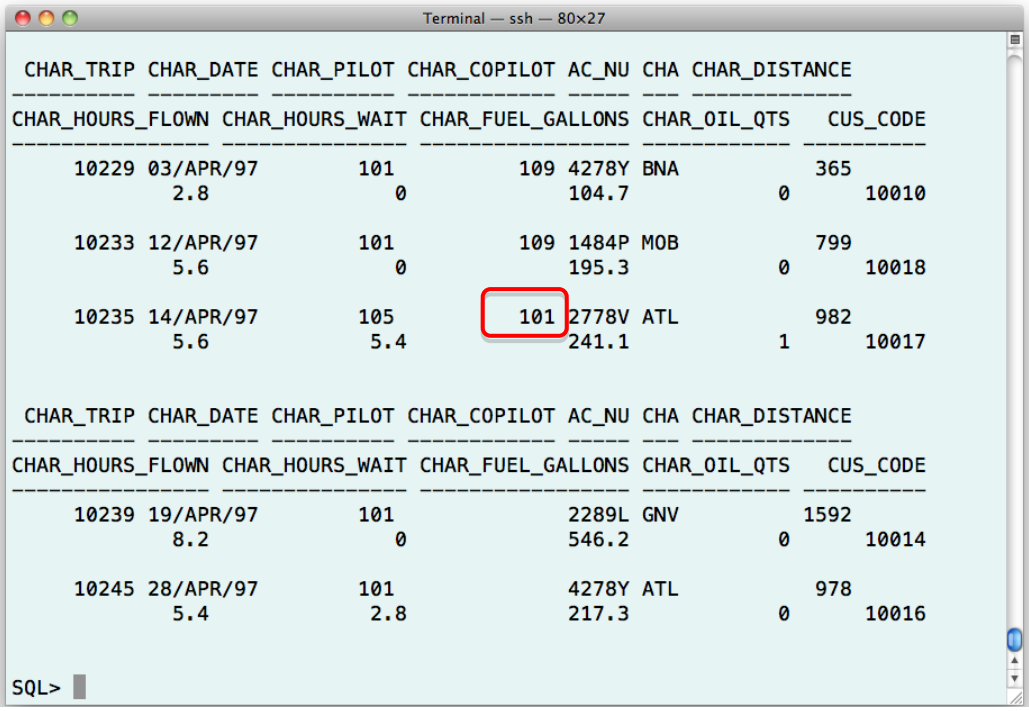
The result is as follows. Look at the three records of pilot 101 in April 1997.




```
------ ---------- ---------- -------------- ---------- ----------
199611 PA23-250          101            6.1      191.5    1721.56
199611 PA31-350          101             16      679.7     5668.2
199612 PA23-250          101            5.6      195.3    1542.07
199612 C-90A             101            8.2      546.2    4229.28
199612 PA31-350          101            5.4      217.3     2298.3
199701 PA31-350          101            9.3      427.3    3673.05
199701 C-90A             101           10.1      669.6    5278.59
199702 PA31-350          101           12.2      489.5    5177.05
199703 C-90A             101            1.8      130.5     899.79
199703 PA23-250          101            6.1      191.5    1721.56
199703 PA31-350          101           13.2        575    4810.45

TIME_I MOD_CODE      EMP_NUM TOT_CHAR_HOURS   TOT_FUEL    REVENUE
------ ---------- ---------- -------------- ---------- ----------
199704 PA23-250          101            5.6      195.3    1542.07
199704 PA31-350          101            8.2        322    3156.05
199704 C-90A             101            8.2      546.2    4250.64
199705 PA31-350          101            9.3      427.3    3673.05
199706 C-90A             101           10.1      669.6    5278.59
199706 PA31-350          101            1.6       72.6        752

94 rows selected.

SQL>
```

Now let's check employee 101 as pilot and copilot in April 1997:

```
select *
from dw.charter
where to_char(char_date, 'YYYYMM') = '199704'
and (char_pilot = 101 or char_copilot = 101);
```



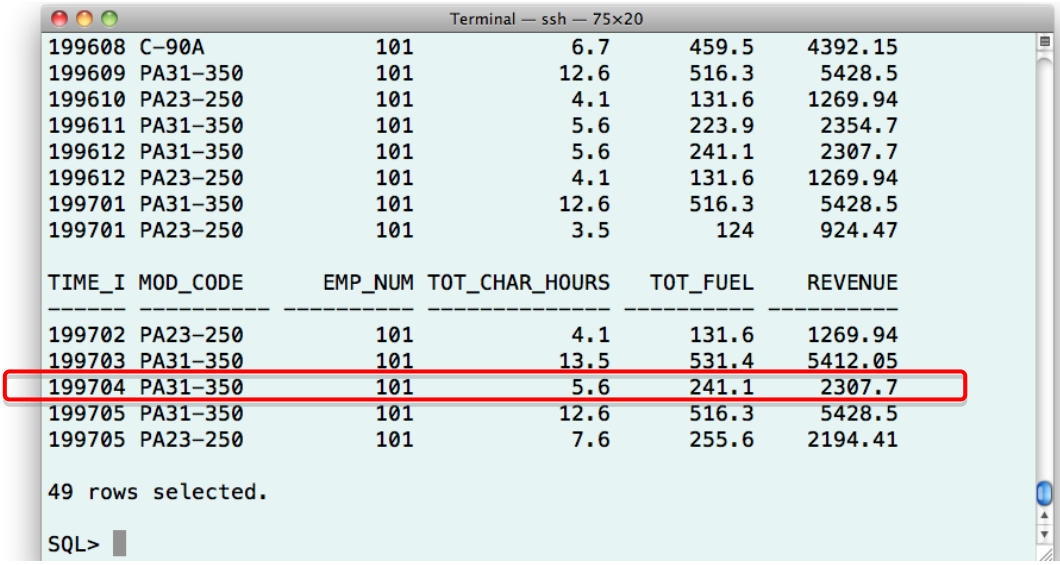
The result shows that employee 101 was a copilot in one charter flight in April 1997.

The fact table for employee 101 does not include the data of employee 101 as a copilot. You can double-check the attributes of tot_char_hours, tot_fuel_used, and revenue in the fact table. One way to solve this problem is to create another fact table solely for copilot. Hence we need Charter_fact2:

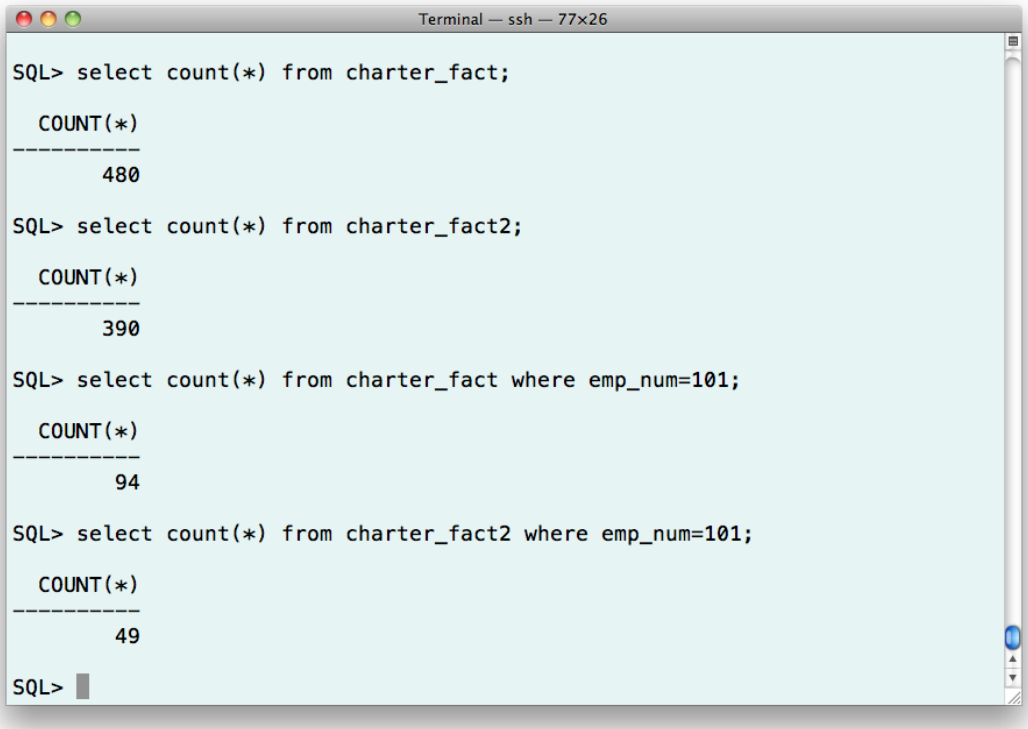
```
create table charter_fact2 as
select C.Char_coPilot as EMP_Num,
       M.Mod_Code,
       to_char(C.Char_Date, 'YYYYMM') as Time_ID,
       sum(C.Char_Hours_Flown) as Tot_Char_Hours,
       sum(C.Char_Fuel_Gallons) as Tot_Fuel,
       sum(C.Char_Distance * M.Mod_chg_mile) as Revenue
from   dw.Charter C, dw.Model M, dw.Aircraft A
where  C.AC_Number=A.AC_Number and A.Mod_Code=M.Mod_Code
group by C.Char_Pilot, M.Mod_Code, to_char(C.Char_Date, 'YYYYMM');
```

Let's see the records of copilot 101:

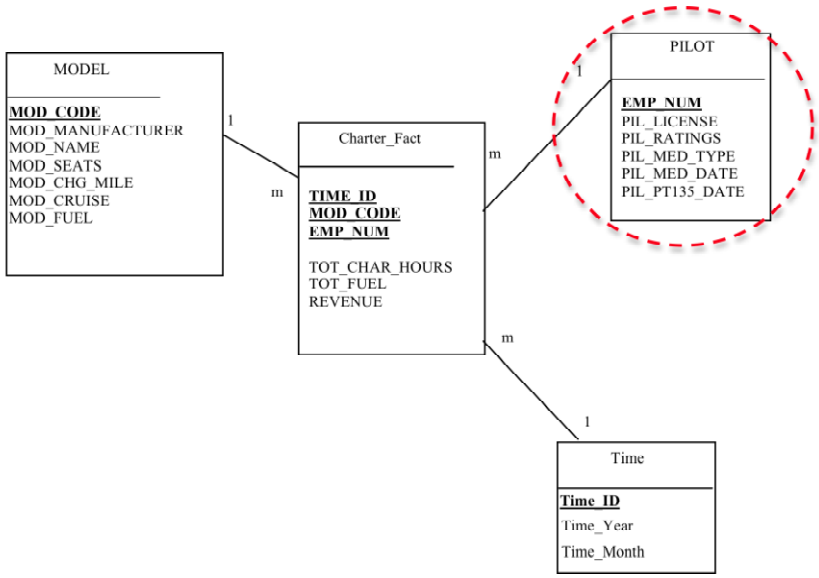
```
select *
from charter_fact2
where emp_num=101
order by time_id;
```


Terminal — ssh — 75×20
199608 C-90A 101 6.7 459.5 4392.15
199609 PA31-350 101 12.6 516.3 5428.5
199610 PA23-250 101 4.1 131.6 1269.94
199611 PA31-350 101 5.6 223.9 2354.7
199612 PA31-350 101 5.6 241.1 2307.7
199612 PA23-250 101 4.1 131.6 1269.94
199701 PA31-350 101 12.6 516.3 5428.5
199701 PA23-250 101 3.5 124 924.47
TIME_I MOD_CODE EMP_NUM TOT_CHAR_HOURS TOT_FUEL REVENUE
199702 PA23-250 101 4.1 131.6 1269.94
199703 PA31-350 101 13.5 531.4 5412.05
199704 PA31-350 101 5.6 241.1 2307.7
199705 PA31-350 101 12.6 516.3 5428.5
199705 PA23-250 101 7.6 255.6 2194.41
49 rows selected.
SQL>

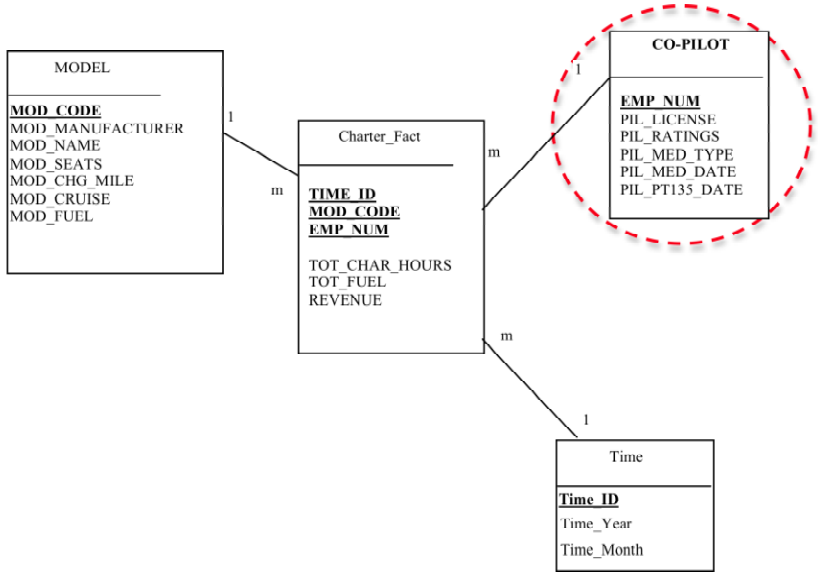
So, now we have two star schemas: one for pilot, and the other for co-pilot. The dimensions and fact for these two star schemas are the same, but the contents of the fact table are different. Let's see first how many records are there in both fact tables.



Star Schema-1 (Pilot)



Star Schema-2 (Co-Pilot):



4. Is it possible to combine the two star schemas (the two facts)?

It is now possible to have a third fact table, which combines both pilot and co-pilot fact measures. So, what we would like to do is to "merge" the two charter facts, so that the new fact table incorporates both pilots and copilots.

To merge the two fact tables, we can use a UNION command in SQL, something like:

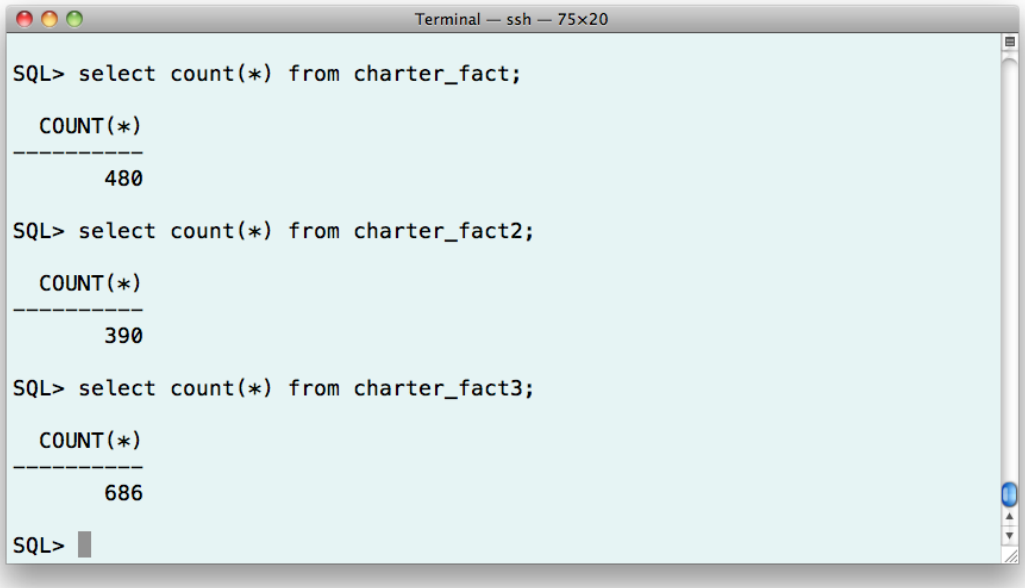
```
select * from charter_fact
union
select * from charter_fact2
```

We can put this UNION command in a CREATE TABLE AS SELECT command as follows:

```
create table charter_fact3 as
select time_id,
       mod_code,
       emp_num,
       sum(tot_char_hours) as tot_char_hours,
       sum(tot_fuel) as tot_fuel,
       sum(revenue) as revenue
from (
   select * from charter_fact
   union
   select * from charter_fact2)
group by time_id, mod_code, emp_num;
```

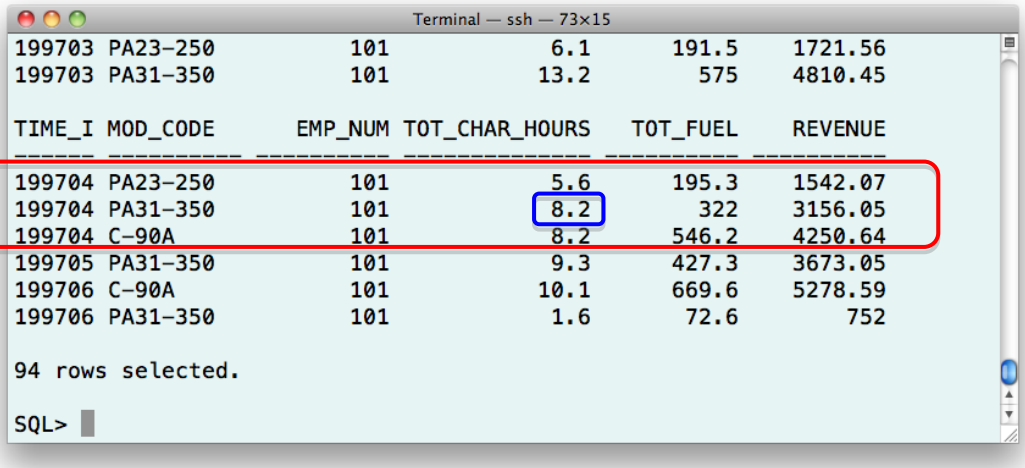
It basically merging the two fact tables, and then the result of this merging or union process is grouped by time_id, mod_code, and emp_num, so that records having the same time_id, mod_code, and emp_num are grouped into one record, where the new sum is then calculated.

The following shows number of records in charter_fact, charter_fact2, and charter_fact3 tables:



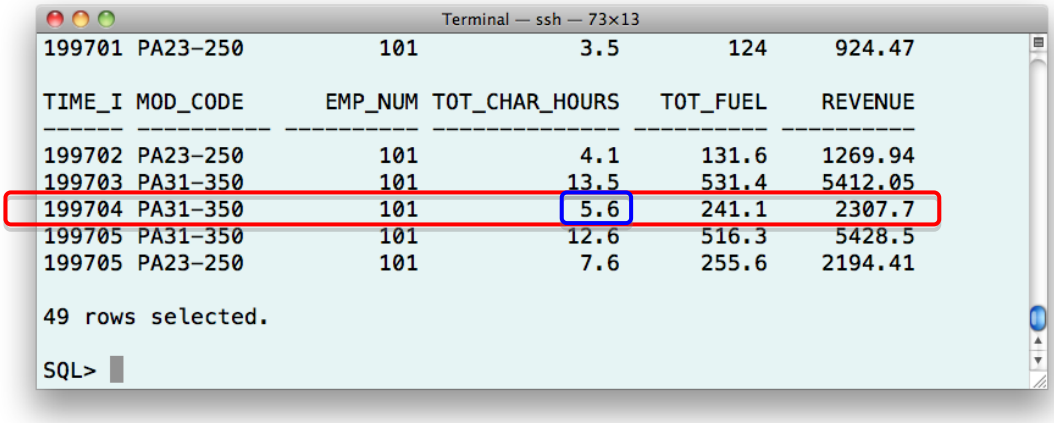
Now, let's check employee 101 as a case study:

```
select *
from charter_fact
where emp_num=101
order by time_id;
```



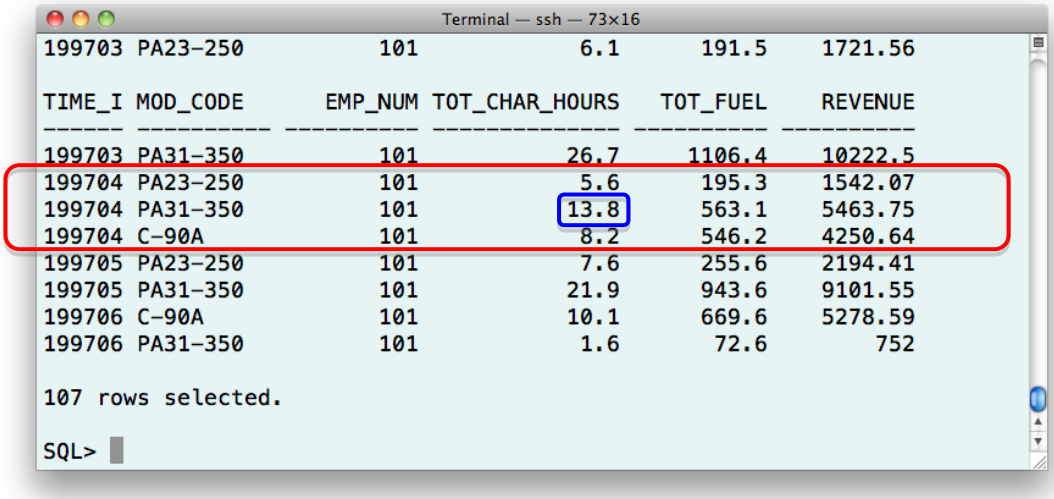
And now employee 101 in charter_fact2 (as a copilot):

```
select *
from charter_fact2
where emp_num=101
order by time_id;
```



And finally, employee 101 in charter_fact3:

```
select *
from charter_fact3
where emp_num=101
order by time_id;
```



Notice that the calculation of tot_char_hours, tot_fuel, and revenue now incorporates both pilot and copilot data.

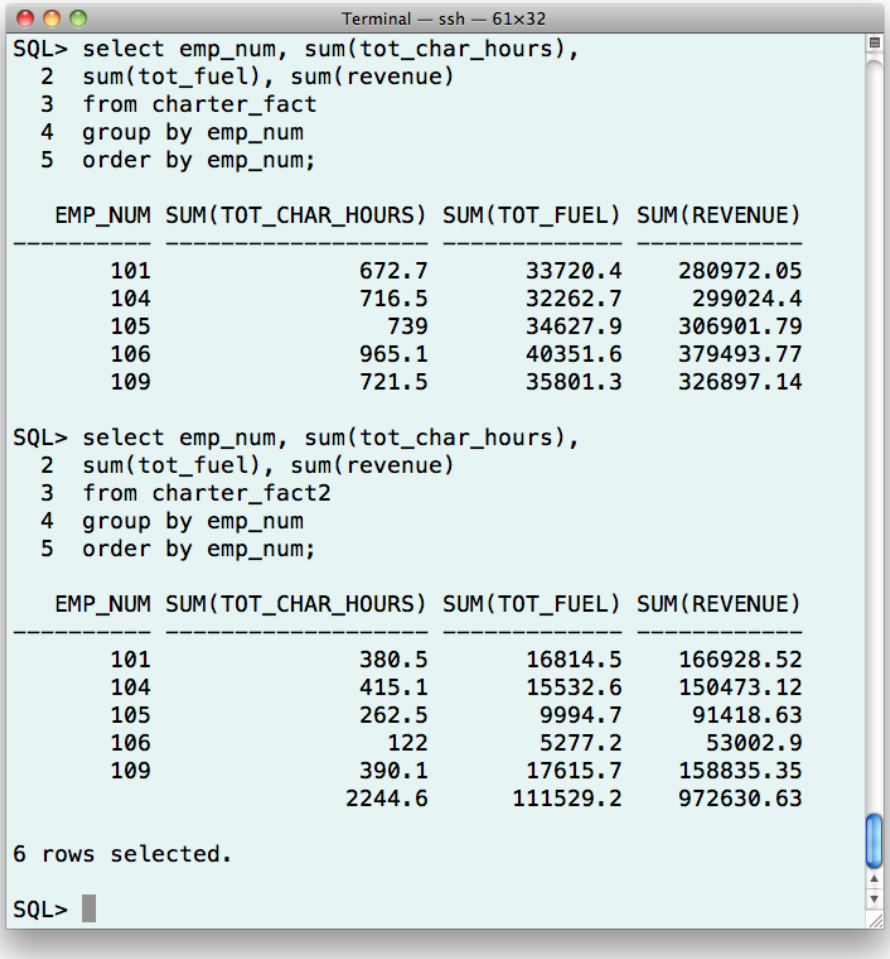
Is this correct?
The answer is No!

Tot_char_hourss in charter_fact3, as shown in the above screenshot, is correct only in the context of Emp_num, which in this case it is pilot and co-pilot. In the above example, Emp_num 101, is a pilot in April 1997 flying aircraft model PA31-350 accumulated 8.2 hours, and the same pilot on the same month flying the same aircraft model as a copilot accumulated 5.6 hours, with a total of 13.8 hours. So, this is correct.

However, if you look at from the month point of view (e.g. April 1997), total hours of 13.8 is a double-dip, because when Emp_num 101 is a co-pilot, the flying hour is also recorded by the pilot of that flight (which is not Emp_num 101). Therefore, it is not possible to just merge the two facts.

However, we are able to merge to two facts, if we only take Pilot/Co-Pilot dimension, and not the other two dimensions.

The following are the fact measures of pilots and co-pilots from the two facts.



Now, if we merge the above two facts, but only taking emp_num dimension, then the result is as follows:

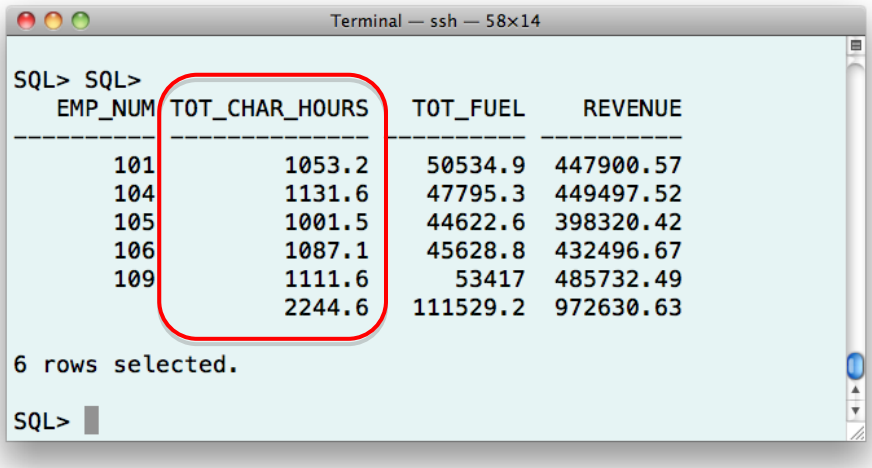
```
create table charter_fact3b as
select emp_num,
       sum(tot_char_hours) as tot_char_hours,
       sum(tot_fuel) as tot_fuel,
       sum(revenue) as revenue
from (
   select * from charter_fact
   union
   select * from charter_fact2)
group by emp_num
order by emp_num;
```

Or you could do this:

```
create table charter_fact3b as
select emp_num,
       sum(tot_char_hours) as tot_char_hours,
       sum(tot_fuel) as tot_fuel,
       sum(revenue) as revenue
from (
   select emp_num,
       sum(tot_char_hours) as tot_char_hours,
       sum(tot_fuel) as tot_fuel,
       sum(revenue) as revenue
   from charter_fact
   group by emp_num
   union
   select emp_num,
       sum(tot_char_hours) as tot_char_hours,
       sum(tot_fuel) as tot_fuel,
       sum(revenue) as revenue
   from charter_fact2
   group by emp_num)
group by emp_num
order by emp_num;
```

What is the difference between the two create table charter_fact3b methods?
The result will still be the same.

```
select * from charter_fact3b;
```



If we look at Total Charter Hours in Fact3, this column shows the total hours for each employee, whether the employee is a pilot or a co-pilot. For example, for Employee 101, the total charter hours is 1053.2 hours (as a pilot or as a co-pilot). This is because we take the data from charter_fact (for pilot), union it with charter_fact2 (for co-pilot), and re-calculate the total charter hours.

However, if we look at the other two fact measures: Total Fuel and Total Revenue, the analysis will be misleading. For example, it is true that Employee 101 (as a pilot

or as a co-pilot) used up 50534.9 gallons of fuel. However, some of the trips that 101 made as a pilot, had someone else as a co-pilot (e.g. Employee 105). But Employee 105 also calculated the trip that he made as a co-pilot in the total fuel calculation; hence, the calculation of the total fuel is doubled; and is therefore incorrect.

The same mistake also happens for the Total Revenue. It is try that Employee 101 (as a pilot and as a co-pilot) brings a total revenue of $447900.57. However, some of the trips that Employee 101 made are together with another employee. As a result, the revenue for that particular trip will be counted twice; and is therefore incorrect.

So, Total Fuel Used and Total Revenue fact measures should be excluded from charter_fact3. This raises two questions:

1. Why is total charter hours correct, whereas total fuel and total revenue are incorrect?
2. Is it still necessary to create charter_fact3 with employee as the only dimension and total charter hours as the only fact measure?

The answer to question 1 is that total charter hours are **individual**. Each employee has his/her own flying hours, and even when they fly together as a pilot and co-pilot, the flying hours are still individual.

On the other hand, total fuel used and total revenue are **not individual** – they are related to the company. So for example, in one trip, that has a pilot and a co-pilot, has $1000 of revenue and use 100 gallons of fuel, it does not mean that the revenue of the pilot is $1000 and the revenue of the co-pilot is another $1000; a total of $2000. The revenue is still $1000, regardless whether there is a pilot only on the trip, or a pilot and a co-pilot on that trip. The same principal is applied to the fuel used.

In order to answer question 2, we need to evaluate how to create charter_fact3 correct. This is the charter_fact3 having emp_num and tot_char_hours only.

```
create table charter_fact3c as
select emp_num, sum(tot_char_hours) as tot_char_hours
from (
   select * from charter_fact
   union
   select * from charter_fact2)
group by emp_num
order by emp_num;
```

However, the use of charter_fact3 is rather limited. We could query the first two charter facts directly, using the following SQL:

```
select emp_num, sum(tot_char_hours) as tot_char_hours
from (
   select * from charter_fact
   union
   select * from charter_fact2)
group by emp_num
order by emp_num;
```

Therefore, the first two facts are sufficient: one star schema for pilot, and another star schema for co-pilot.

The main reason why it is not necessary to create a combined fact, is because the transaction (e.g. a charter trip), can have two “personnel”. Therefore, we need to have two star schema to capture the roles of these two personnel: pilot and co-pilot. Combining two star schemas into one is generally not possible or with a limited use.

On the other hand, if the transaction records have one personnel or pilot (for example, there is no notion of co-pilot), but there are two types of pilot: permanent pilot and sessional pilot; if we have two star schemas: one for permanent pilot and another for sessional pilot, these two star schemas can be combined by using a simple union operator, without any re-calculation of the fact measures. Because, for example, the total revenue of a permanent pilot will not overlap with the total revenue of a sessional pilot. Combining these two records in a new fact will remain two records.