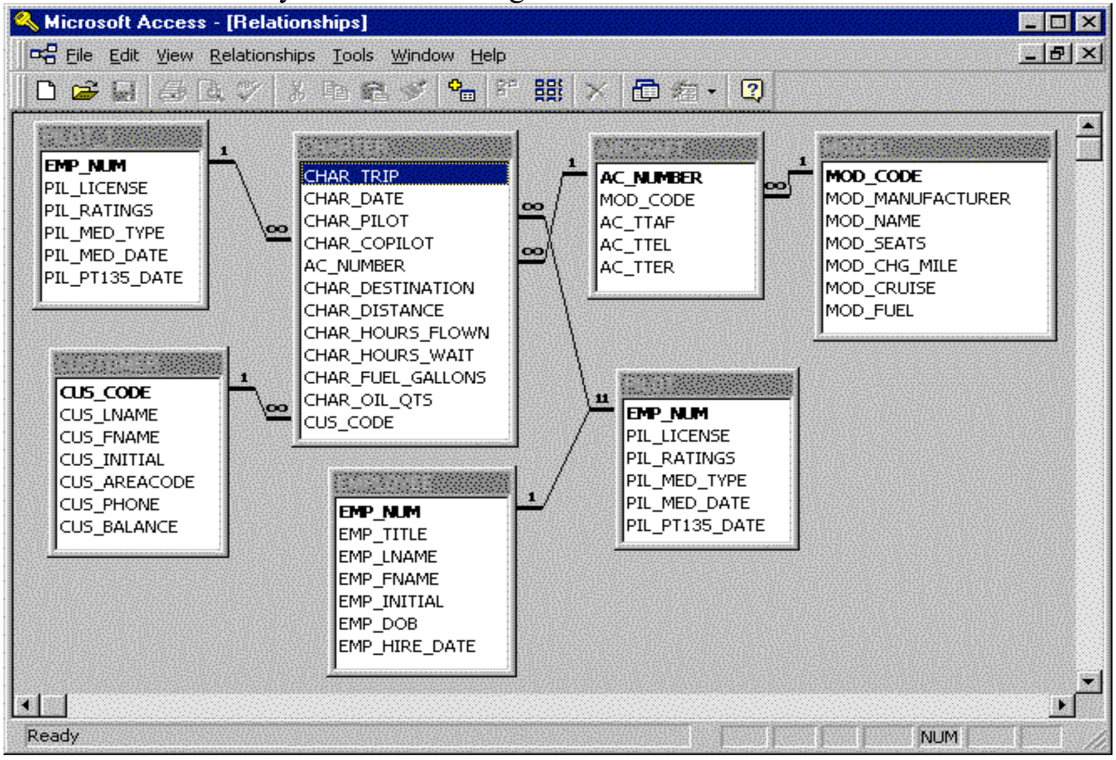
Data Exploration
The Robcor Case Study

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:



Microsoft Access - [Relationships]
File Edit View Relationships Tools Window Help
Ready
NUM

```
EMP_NUM
PIL_LICENSE
PIL_RATINGS
PIL_MED_TYPE
PIL_MED_DATE
PIL_PT135_DATE

CHAR_TRIP
CHAR_DATE
CHAR_PILOT
CHAR_COPILOT
AC_NUMBER
CHAR_DESTINATION
CHAR_DISTANCE
CHAR_HOURS_FLOWN
CHAR_HOURS_WAIT
CHAR_FUEL_GALLONS
CHAR_OIL_QTS
CUS_CODE

AC_NUMBER
MOD_CODE
AC_TTAF
AC_TTEL
AC_TTER

MOD_CODE
MOD_MANUFACTURER
MOD_NAME
MOD_SEATS
MOD_CHG_MILE
MOD_CRUISE
MOD_FUEL

CUS_CODE
CUS_LNAME
CUS_FNAME
CUS_INITIAL
CUS_AREACODE
CUS_PHONE
CUS_BALANCE

EMP_NUM
PIL_LICENSE
PIL_RATINGS
PIL_MED_TYPE
PIL_MED_DATE
PIL_PT135_DATE

EMP_NUM
EMP_TITLE
EMP_LNAME
EMP_FNAME
EMP_INITIAL
EMP_DOB
EMP_HIRE_DATE
```

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 …
```

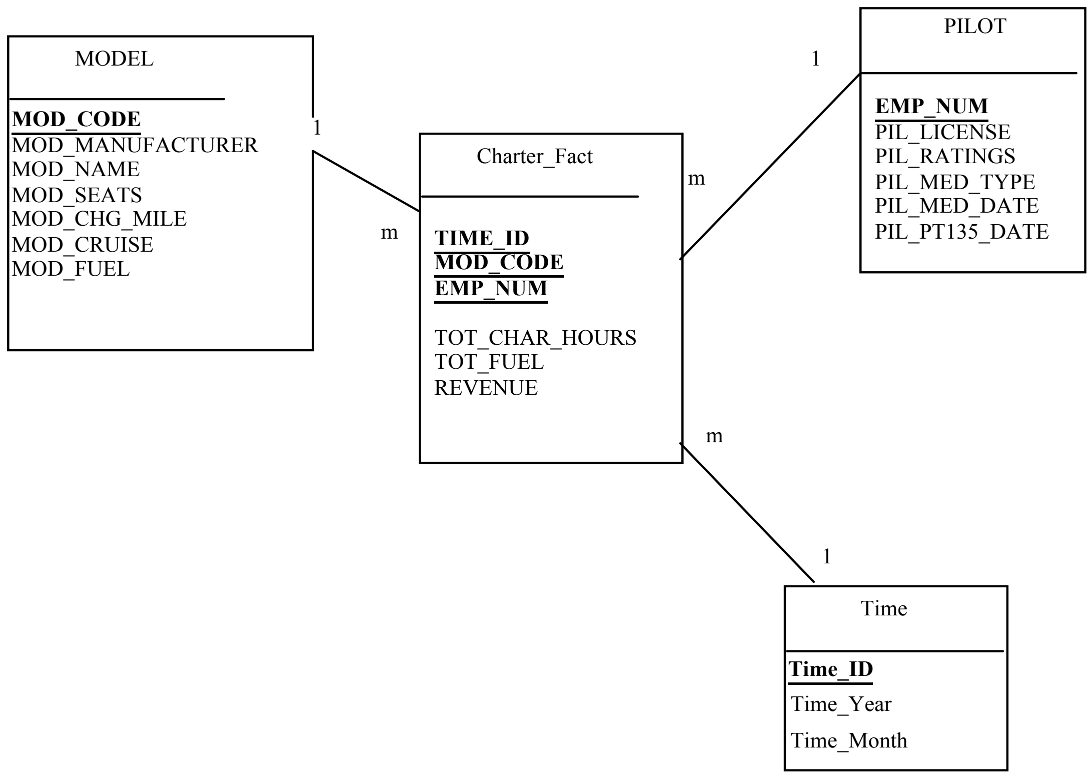
Given these requirements, complete the following:

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



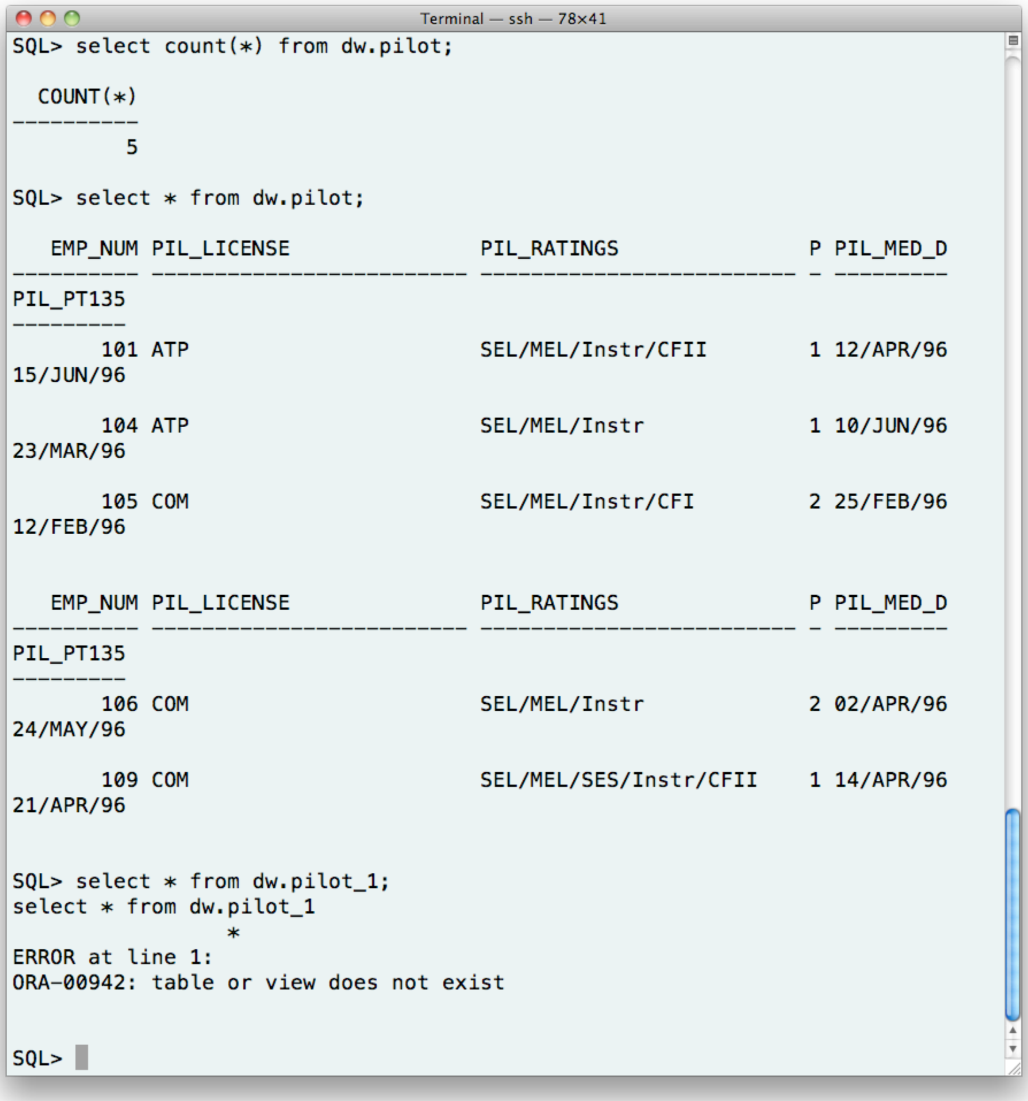
```
--First create the dimensions
create table time As
select Distinct to_char(char_date, 'YYYYMM') as Time_ID,
                to_char(char_date, 'Month') as Time_Month,
                to_char(char_date, 'YYYY') as Time_Year
from dw.Charter;

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:

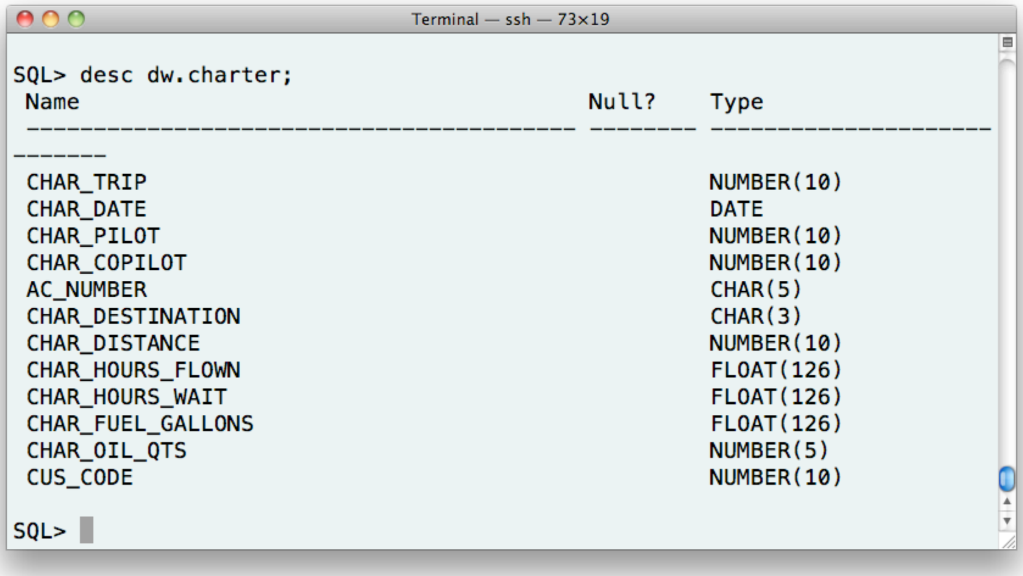
1. Where is Pilot_1 table?

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.



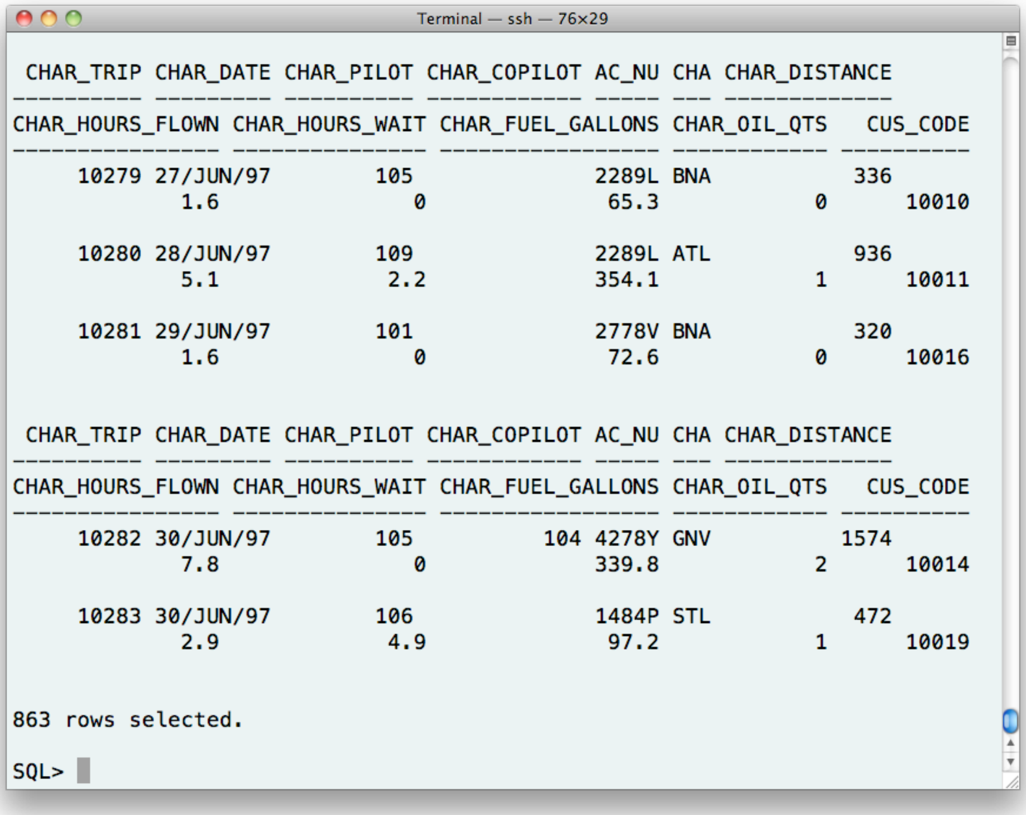
```
SQL> select count(*) from dw.pilot;

  COUNT(*)
----------
         5

SQL> select * from dw.pilot;

   EMP_NUM PIL_LICENSE                PIL_RATINGS                P PIL_MED_D
---------- -------------------------- -------------------------- - ---------
PIL_PT135
---------
       101 ATP                        SEL/MEL/Instr/CFII         1 12/APR/96
15/JUN/96

       104 ATP                        SEL/MEL/Instr              1 10/JUN/96
23/MAR/96

       105 COM                        SEL/MEL/Instr/CFI          2 25/FEB/96
12/FEB/96


   EMP_NUM PIL_LICENSE                PIL_RATINGS                P PIL_MED_D
---------- -------------------------- -------------------------- - ---------
PIL_PT135
---------
       106 COM                        SEL/MEL/Instr              2 02/APR/96
24/MAY/96

       109 COM                        SEL/MEL/SES/Instr/CFII     1 14/APR/96
21/APR/96


SQL> select * from dw.pilot_1;
select * from dw.pilot_1
                 *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> 
```

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



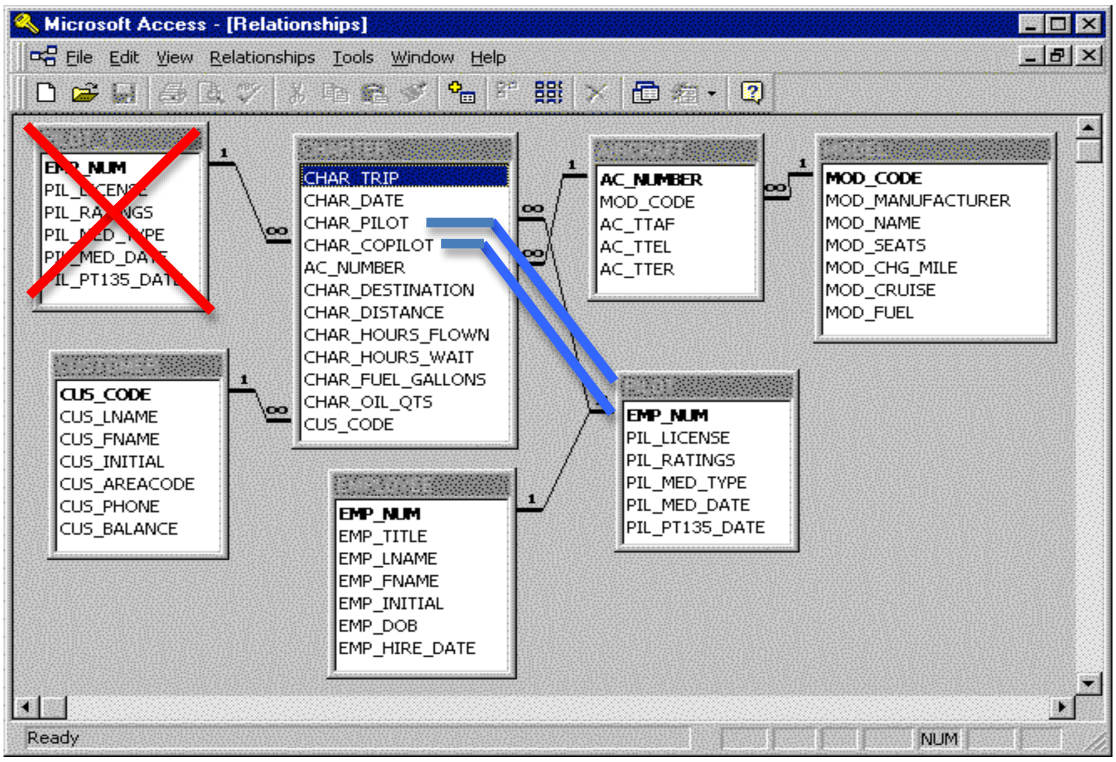
```
SQL> desc dw.charter;
 Name                                      Null?    Type
 ----------------------------------------- -------- ---------------------
-------
 CHAR_TRIP                                          NUMBER(10)
 CHAR_DATE                                          DATE
 CHAR_PILOT                                         NUMBER(10)
 CHAR_COPILOT                                       NUMBER(10)
 AC_NUMBER                                          CHAR(5)
 CHAR_DESTINATION                                   CHAR(3)
 CHAR_DISTANCE                                      NUMBER(10)
 CHAR_HOURS_FLOWN                                   FLOAT(126)
 CHAR_HOURS_WAIT                                    FLOAT(126)
 CHAR_FUEL_GALLONS                                  FLOAT(126)
 CHAR_OIL_QTS                                       NUMBER(5)
 CUS_CODE                                           NUMBER(10)

SQL> 
```

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



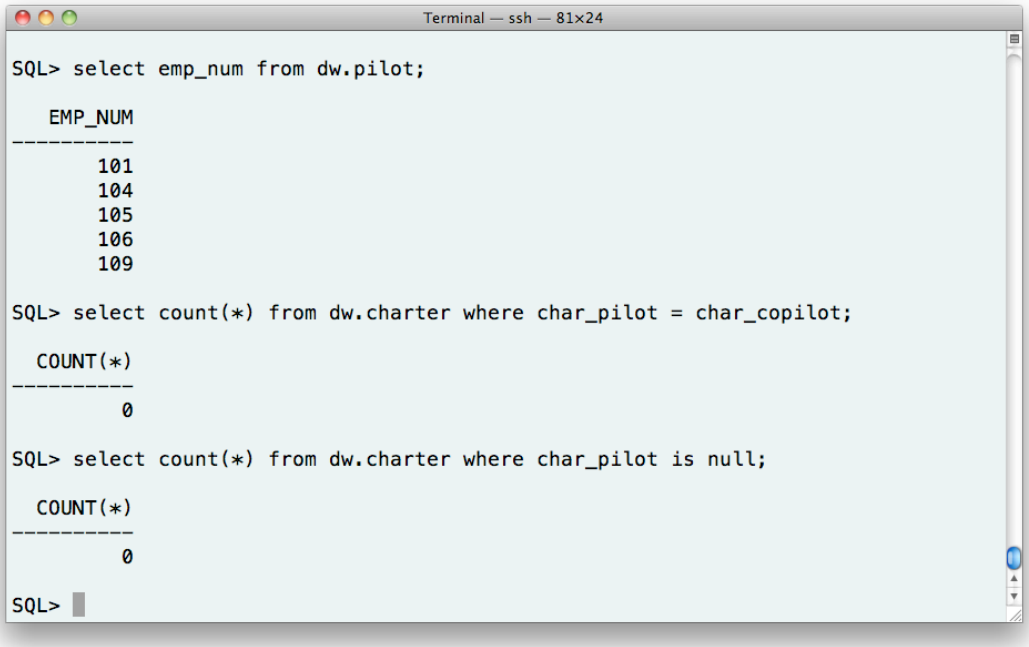
```
 CHAR_TRIP CHAR_DATE CHAR_PILOT CHAR_COPILOT AC_NU CHA CHAR_DISTANCE
---------- --------- ---------- ------------ ----- --- -------------
CHAR_HOURS_FLOWN CHAR_HOURS_WAIT CHAR_FUEL_GALLONS CHAR_OIL_QTS   CUS_CODE
---------------- --------------- ----------------- ------------ ----------
     10279 27/JUN/97        105              2289L BNA           336
        1.6               0              65.3            0      10010

     10280 28/JUN/97        109              2289L ATL           936
        5.1             2.2             354.1            1      10011

     10281 29/JUN/97        101              2778V BNA           320
        1.6               0              72.6            0      10016


 CHAR_TRIP CHAR_DATE CHAR_PILOT CHAR_COPILOT AC_NU CHA CHAR_DISTANCE
---------- --------- ---------- ------------ ----- --- -------------
CHAR_HOURS_FLOWN CHAR_HOURS_WAIT CHAR_FUEL_GALLONS CHAR_OIL_QTS   CUS_CODE
---------------- --------------- ----------------- ------------ ----------
     10282 30/JUN/97        105          104 4278Y GNV          1574
        7.8               0             339.8            2      10014

     10283 30/JUN/97        106              1484P STL           472
        2.9             4.9              97.2            1      10019


863 rows selected.

SQL> 
```

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:



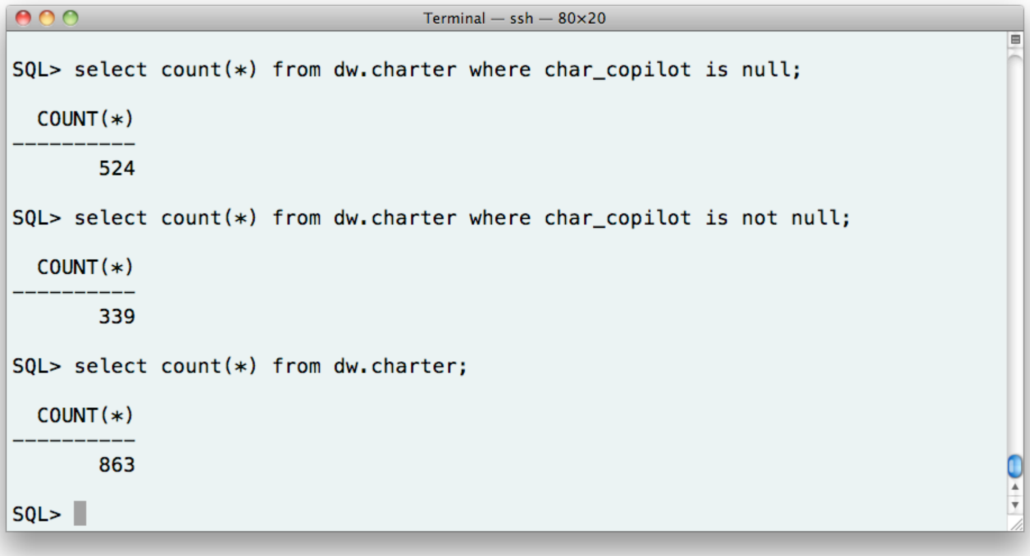
While on the subject of pilot and copilot, let's explore more:

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.



```
SQL> select emp_num from dw.pilot;

   EMP_NUM
----------
       101
       104
       105
       106
       109

SQL> select count(*) from dw.charter where char_pilot = char_copilot;

  COUNT(*)
----------
         0

SQL> select count(*) from dw.charter where char_pilot is null;

  COUNT(*)
----------
         0

SQL> 
```

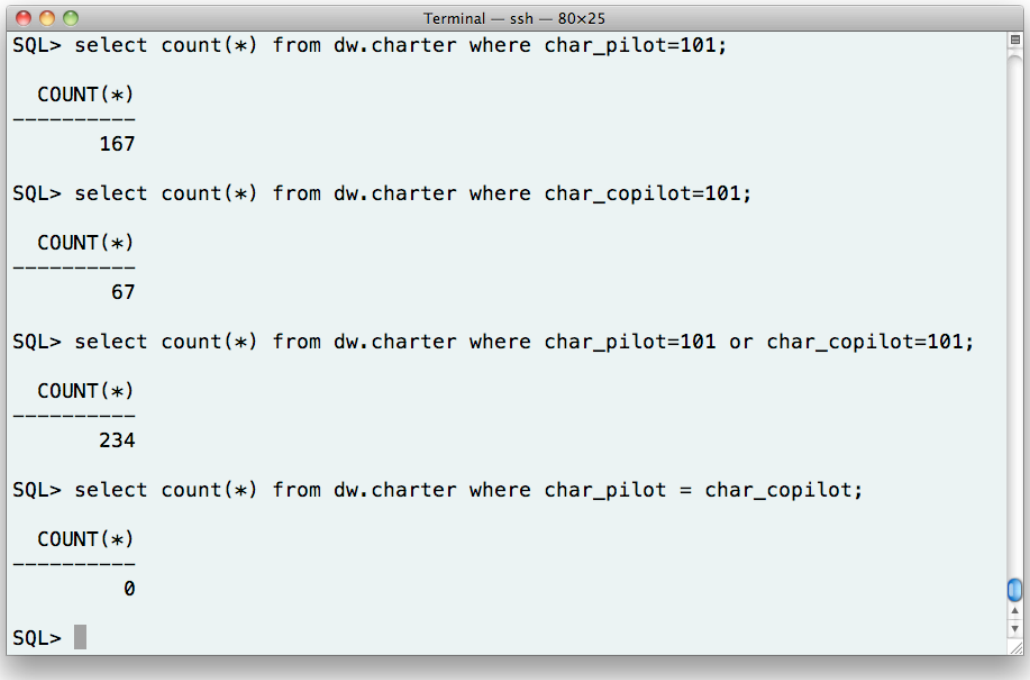
Now, let's check the copilots. We know that not all flights have copilots. Let's check how many flights with copilots, and how many flights without copilots, and see whether the sum of these match with the total number of records in dw.charter.



```
SQL> select count(*) from dw.charter where char_copilot is null;

  COUNT(*)
----------
       524

SQL> select count(*) from dw.charter where char_copilot is not null;

  COUNT(*)
----------
       339

SQL> select count(*) from dw.charter;

  COUNT(*)
----------
       863

SQL> 
```

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

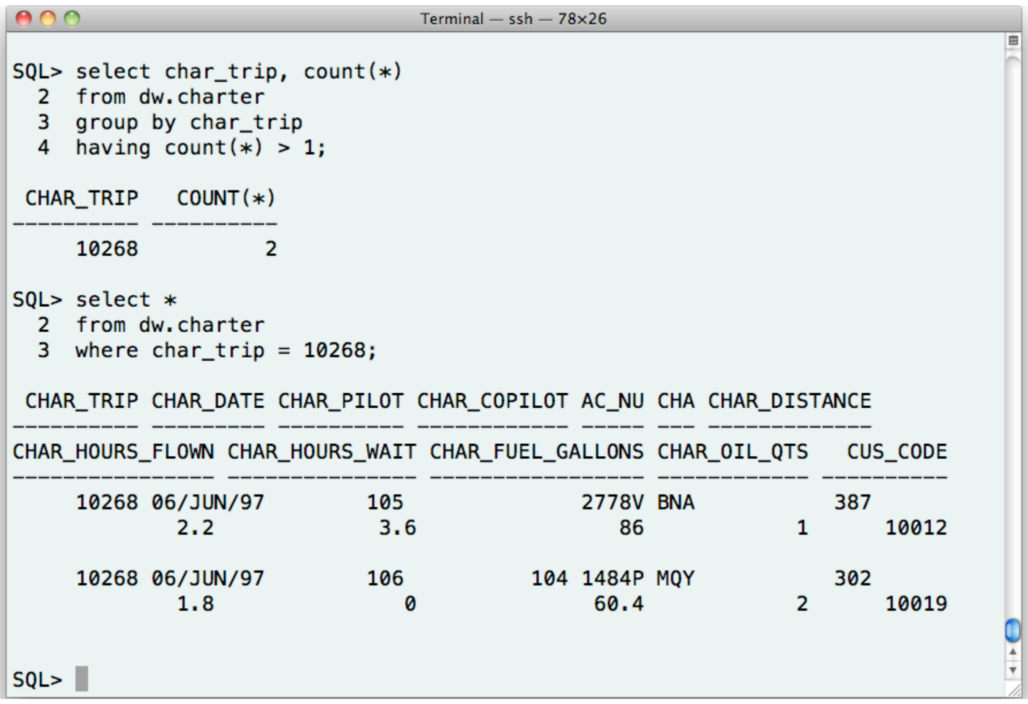
The following shows number of charter flights for pilot 101 as a pilot, and as a copilot. The sum shows that, in this case, employee 101 cannot be pilot and a copilot for the same charter record. So, this is correct.



```
SQL> select count(*) from dw.charter where char_pilot=101;

  COUNT(*)
----------
       167

SQL> select count(*) from dw.charter where char_copilot=101;

  COUNT(*)
----------
        67

SQL> select count(*) from dw.charter where char_pilot=101 or char_copilot=101;

  COUNT(*)
----------
       234

SQL> select count(*) from dw.charter where char_pilot = char_copilot;

  COUNT(*)
----------
         0

SQL> 
```

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.



Terminal — ssh — 78×26

```
SQL> select char_trip, count(*)
  2  from dw.charter
  3  group by char_trip
  4  having count(*) > 1;

 CHAR_TRIP   COUNT(*)
---------- ----------
     10268          2

SQL> select *
  2  from dw.charter
  3  where char_trip = 10268;

 CHAR_TRIP CHAR_DATE CHAR_PILOT CHAR_COPILOT AC_NU CHA CHAR_DISTANCE
---------- --------- ---------- ------------ ----- --- -------------
CHAR_HOURS_FLOWN CHAR_HOURS_WAIT CHAR_FUEL_GALLONS CHAR_OIL_QTS   CUS_CODE
---------------- --------------- ----------------- ------------ ----------
     10268 06/JUN/97        105              2778V BNA           387
             2.2             3.6                86            1      10012

     10268 06/JUN/97        106          104 1484P MQY           302
             1.8               0              60.4            2      10019


SQL> 
```

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

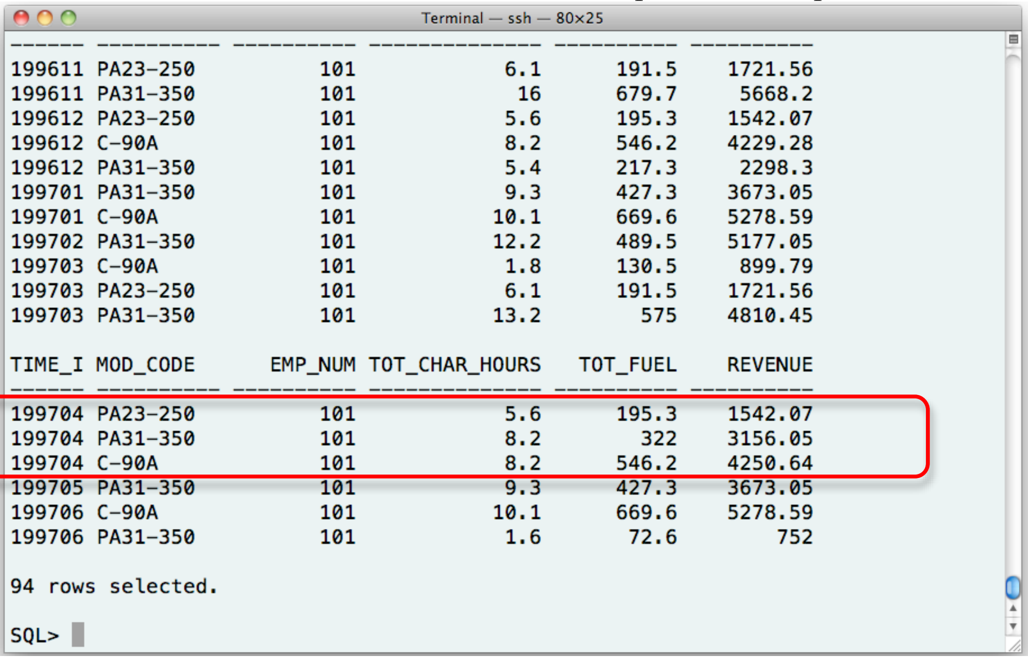
When Charter_fact is created, we do not consider copilot. The SQL to create Charter_fact is as follows:

```
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;
```

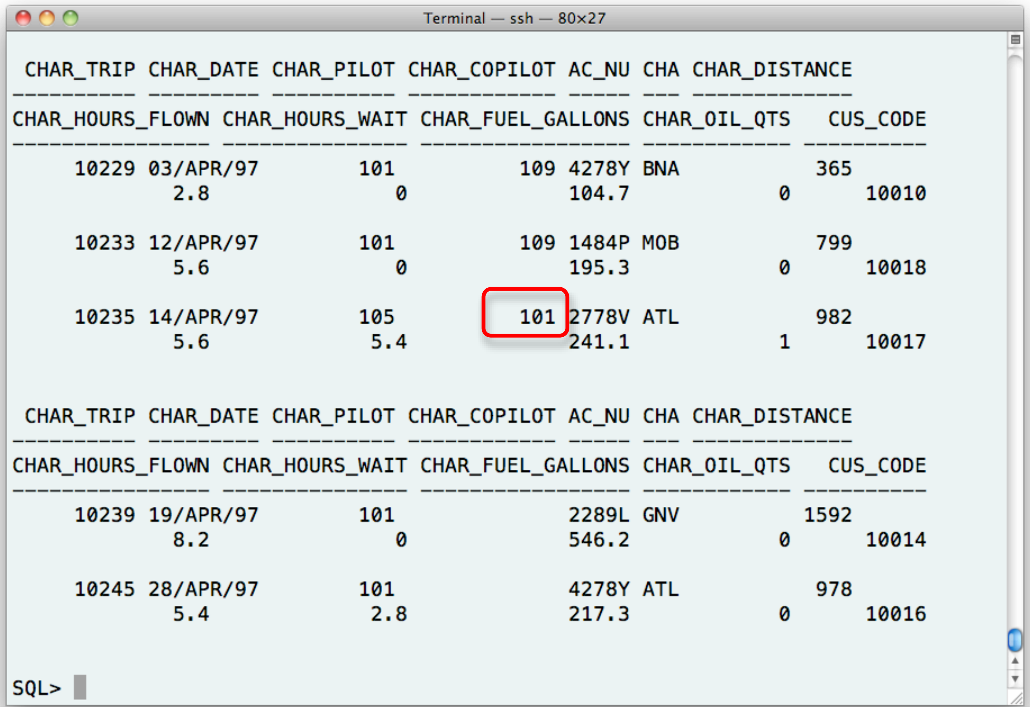
The result is as follows. Look at the three records of pilot 101 in April 1997.



```
------ ---------- ---------- -------------- ---------- ----------
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);
```



```
 CHAR_TRIP CHAR_DATE CHAR_PILOT CHAR_COPILOT AC_NU CHA CHAR_DISTANCE
---------- --------- ---------- ------------ ----- --- -------------
CHAR_HOURS_FLOWN CHAR_HOURS_WAIT CHAR_FUEL_GALLONS CHAR_OIL_QTS   CUS_CODE
---------------- --------------- ----------------- ------------ ----------
     10229 03/APR/97        101          109 4278Y BNA           365
             2.8               0             104.7            0      10010

     10233 12/APR/97        101          109 1484P MOB           799
             5.6               0             195.3            0      10018

     10235 14/APR/97        105          101 2778V ATL           982
             5.6             5.4             241.1            1      10017


 CHAR_TRIP CHAR_DATE CHAR_PILOT CHAR_COPILOT AC_NU CHA CHAR_DISTANCE
---------- --------- ---------- ------------ ----- --- -------------
CHAR_HOURS_FLOWN CHAR_HOURS_WAIT CHAR_FUEL_GALLONS CHAR_OIL_QTS   CUS_CODE
---------------- --------------- ----------------- ------------ ----------
     10239 19/APR/97        101              2289L GNV          1592
             8.2               0             546.2            0      10014

     10245 28/APR/97        101              4278Y ATL           978
             5.4             2.8             217.3            0      10016


SQL> 
```

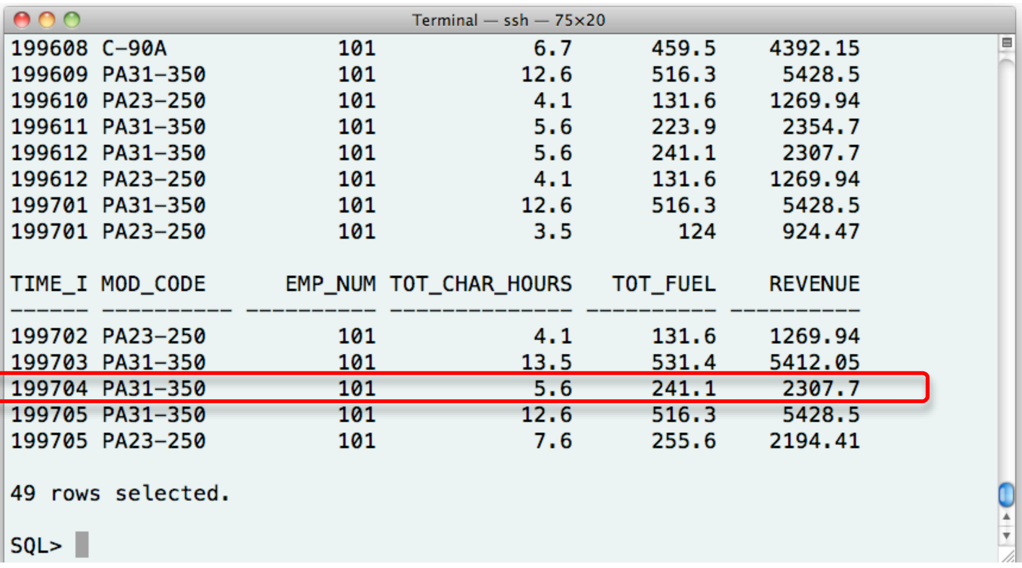
The result shows that employee 101 was a copilot in one charter flight in April 1997.

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');
```

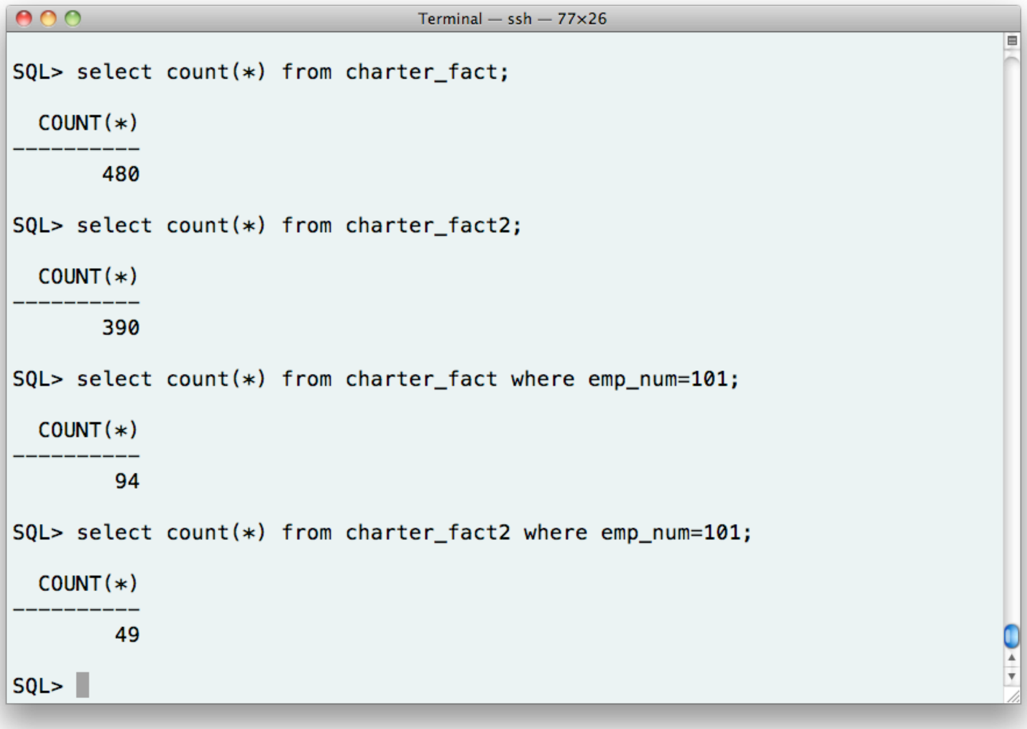
Let's see the records of copilot 101:

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



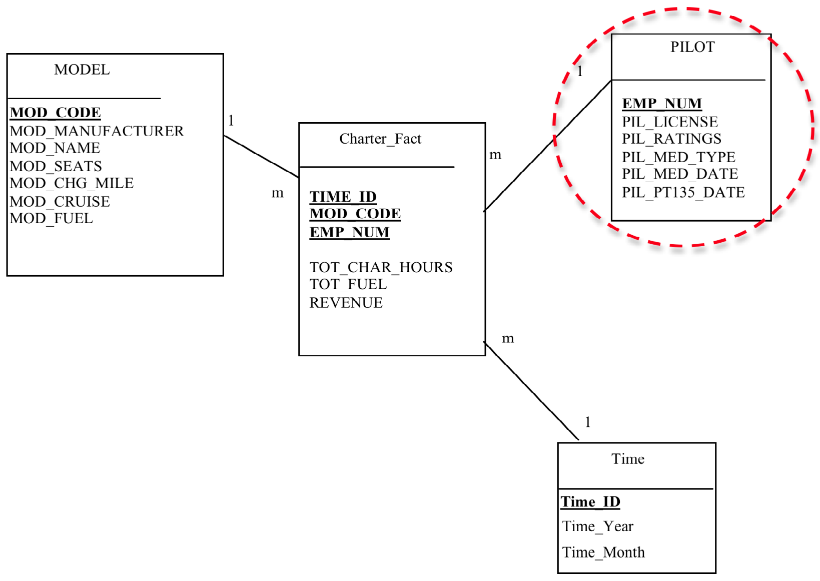
```
199608 C-90A             101            6.7      459.5    4392.15
199609 PA31-350          101           12.6      516.3     5428.5
199610 PA23-250          101            4.1      131.6    1269.94
199611 PA31-350          101            5.6      223.9     2354.7
199612 PA31-350          101            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.

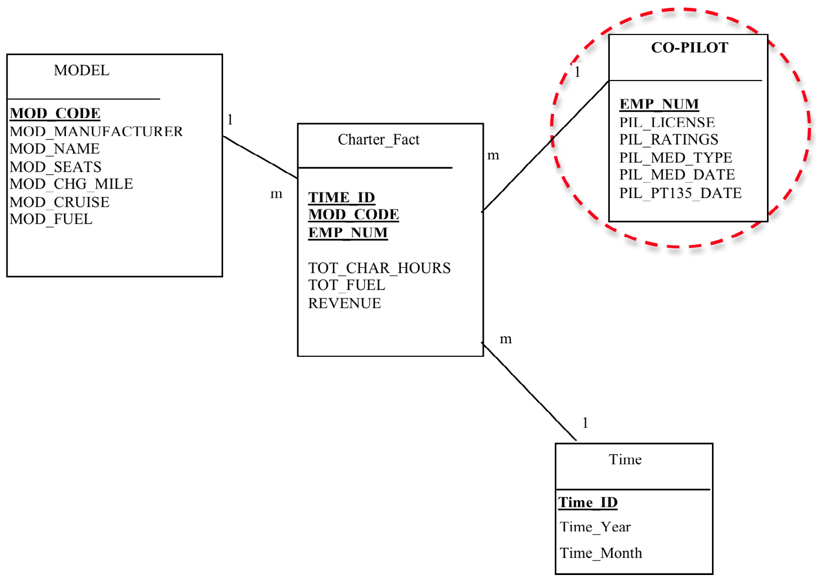


```
SQL> select count(*) from charter_fact;

  COUNT(*)
----------
       480

SQL> select count(*) from charter_fact2;

  COUNT(*)
----------
       390

SQL> select count(*) from charter_fact where emp_num=101;

  COUNT(*)
----------
        94

SQL> select count(*) from charter_fact2 where emp_num=101;

  COUNT(*)
----------
        49

SQL>
```

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:

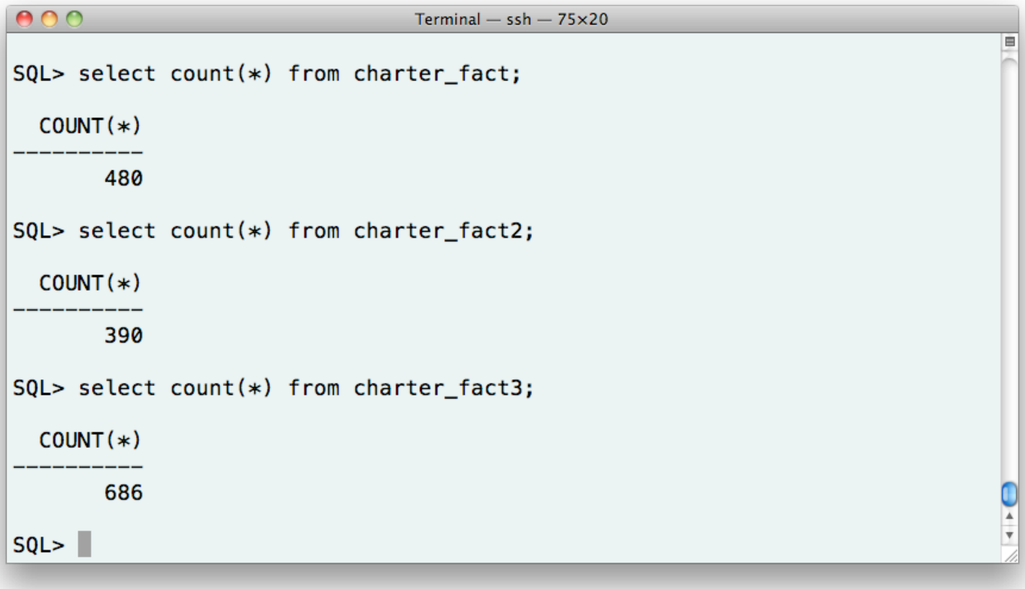
```
select * from charter_fact
union
select * from charter_fact2
```

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.

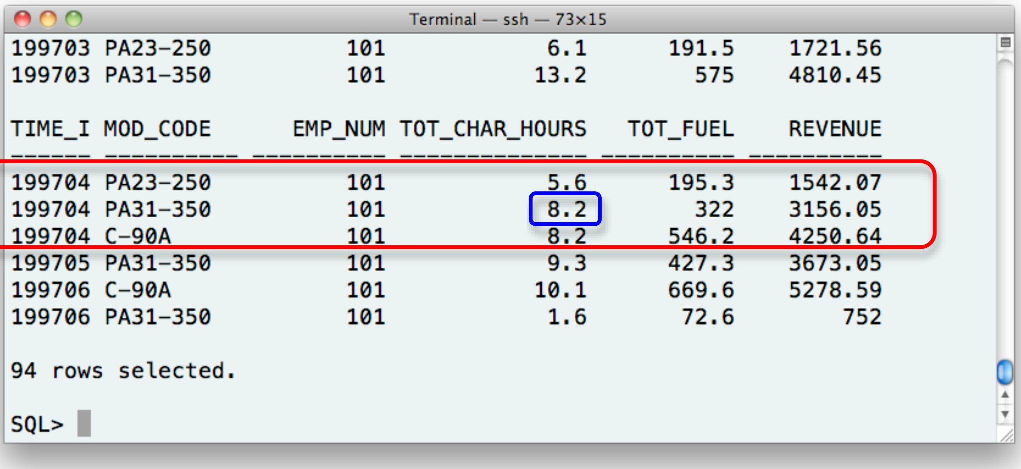
The following shows number of records in charter_fact, charter_fact2, and charter_fact3 tables:



```
SQL> select count(*) from charter_fact;

  COUNT(*)
----------
       480

SQL> select count(*) from charter_fact2;

  COUNT(*)
----------
       390

SQL> select count(*) from charter_fact3;

  COUNT(*)
----------
       686

SQL>
```

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

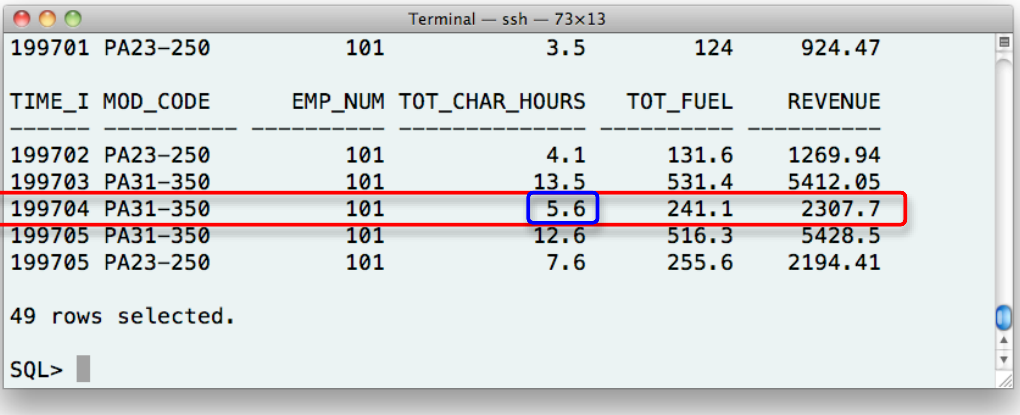
```
select *
from charter_fact
where emp_num=101
order by time_id;
```



```
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>
```

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

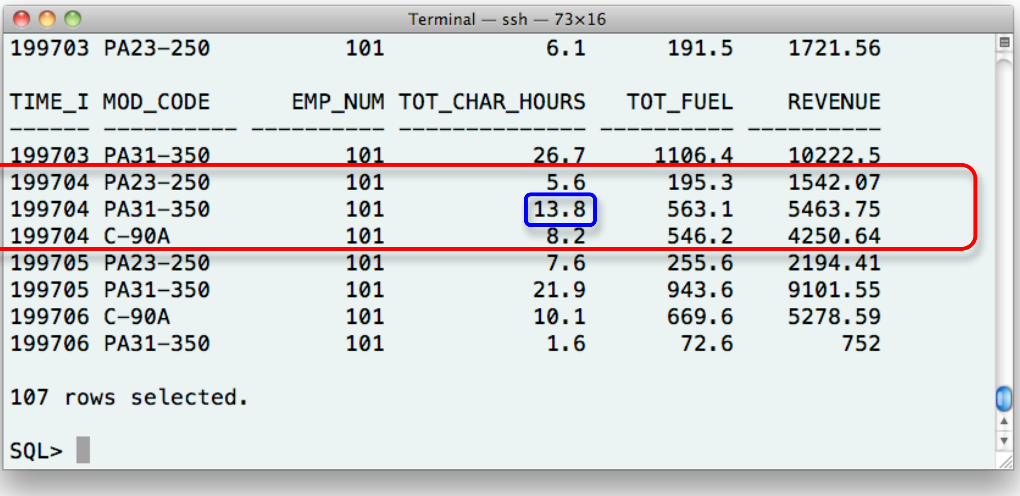
```
select *
from charter_fact2
where emp_num=101
order by time_id;
```



```
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>
```

And finally, employee 101 in charter_fact3:

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



```
199703 PA23-250          101            6.1      191.5    1721.56

TIME_I MOD_CODE       EMP_NUM TOT_CHAR_HOURS   TOT_FUEL    REVENUE
------ ---------- ---------- -------------- ---------- ----------
199703 PA31-350          101           26.7     1106.4    10222.5
199704 PA23-250          101            5.6      195.3    1542.07
199704 PA31-350          101           13.8      563.1    5463.75
199704 C-90A             101            8.2      546.2    4250.64
199705 PA23-250          101            7.6      255.6    2194.41
199705 PA31-350          101           21.9      943.6    9101.55
199706 C-90A             101           10.1      669.6    5278.59
199706 PA31-350          101            1.6       72.6        752

107 rows selected.

SQL>
```

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

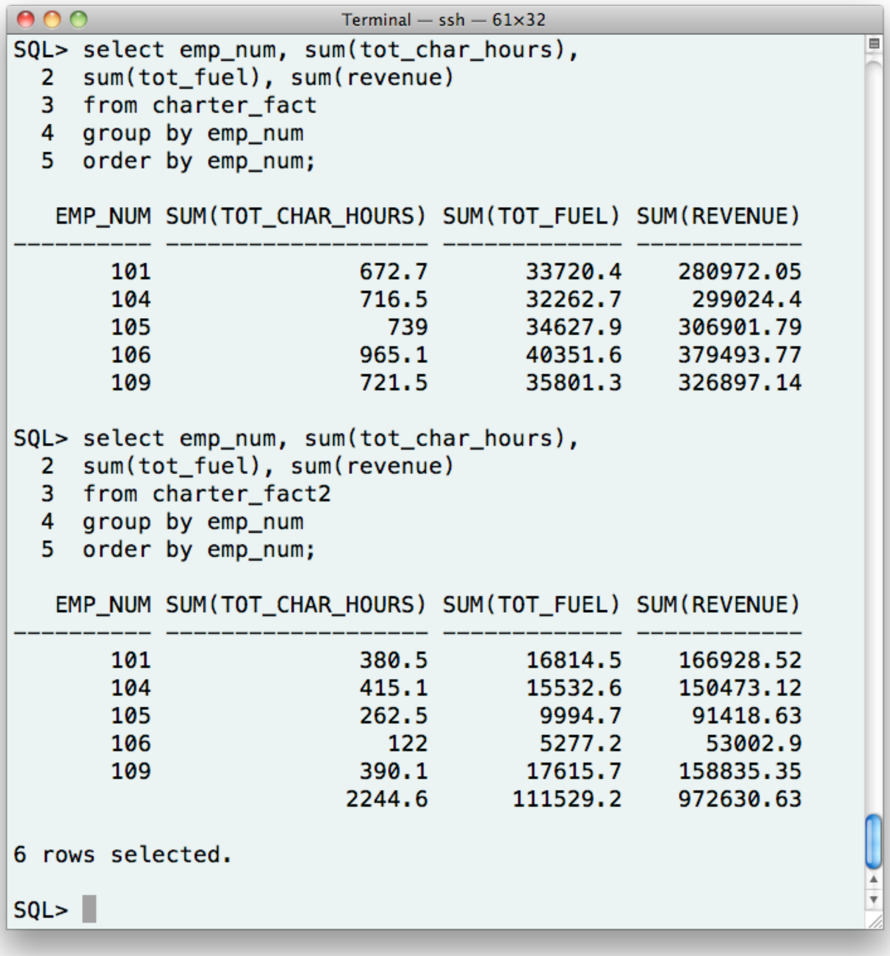
Is this correct?
The answer is No!

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.



```
SQL> select emp_num, sum(tot_char_hours),
  2  sum(tot_fuel), sum(revenue)
  3  from charter_fact
  4  group by emp_num
  5  order by emp_num;

   EMP_NUM SUM(TOT_CHAR_HOURS) SUM(TOT_FUEL) SUM(REVENUE)
---------- ------------------- ------------- ------------
       101               672.7       33720.4    280972.05
       104               716.5       32262.7     299024.4
       105                 739       34627.9    306901.79
       106               965.1       40351.6    379493.77
       109               721.5       35801.3    326897.14

SQL> select emp_num, sum(tot_char_hours),
  2  sum(tot_fuel), sum(revenue)
  3  from charter_fact2
  4  group by emp_num
  5  order by emp_num;

   EMP_NUM SUM(TOT_CHAR_HOURS) SUM(TOT_FUEL) SUM(REVENUE)
---------- ------------------- ------------- ------------
       101               380.5       16814.5    166928.52
       104               415.1       15532.6    150473.12
       105               262.5        9994.7     91418.63
       106                 122        5277.2      53002.9
       109               390.1       17615.7    158835.35
                        2244.6      111529.2    972630.63

6 rows selected.

SQL> 
```

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

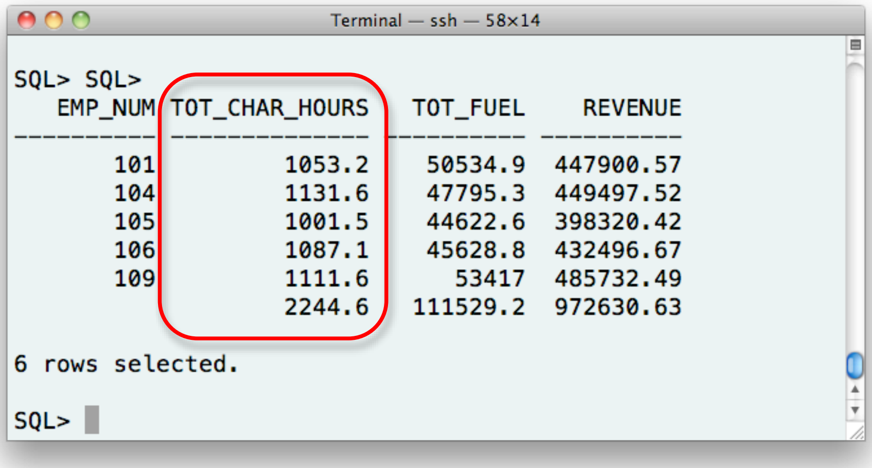
```
create table charter_fact3b as
select emp_num,
       sum(tot_char_hours) as tot_char_hours,
       sum(tot_fuel) as tot_fuel,
       sum(revenue) as revenue
from (
   select * from charter_fact
   union
   select * from charter_fact2)
group by emp_num
order by emp_num;
```

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;
```



```
SQL> SQL>
   EMP_NUM TOT_CHAR_HOURS   TOT_FUEL    REVENUE
---------- -------------- ---------- ----------
       101         1053.2    50534.9  447900.57
       104         1131.6    47795.3  449497.52
       105         1001.5    44622.6  398320.42
       106         1087.1    45628.8  432496.67
       109         1111.6      53417  485732.49
                   2244.6   111529.2  972630.63

6 rows selected.

SQL>
```

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.