Data Cleaning
(The USELOG Case Study)

Description

University Computer Lab's director keeps track of the lab usage, measured by the number of students using the lab. This particular function is very important for budgeting purposes. The computer lab director assigns you the task of developing a small Data Warehouse in which to keep track of the lab usage statistics. The main requirements for this database are to:

a. Show the usage numbers by different time periods (e.g. morning, afternoon, night)

b. Show the usage numbers by time period, by major, and by student's class

c. Compare the usage numbers for different majors and semesters (e.g semester 1 and semester 2).

Use the provided database that includes the following tables: USELOG, STUDENT, MAJOR, and CLASS

USELOG contains the student access data

USELOG (Log_Date, Log_Time, Student_ID, Act)

STUDENT is a table containing student data

STUDENT (Student_ID, Sex, Full/Part, Type, Class_ID, Major_Code)

MAJOR is a table containing major data

MAJOR (Major_Name, Major_Code)

CLASS is a table containing class data

CLASS (Class_Description, Class_ID)

Tasks

The above case study has been discussed in the lecture this week. Your task today is to implement the data warehouse in Oracle.

The following operational databases have been provided for you:

`dw.Class`: table that stores information about classification ids and descriptions

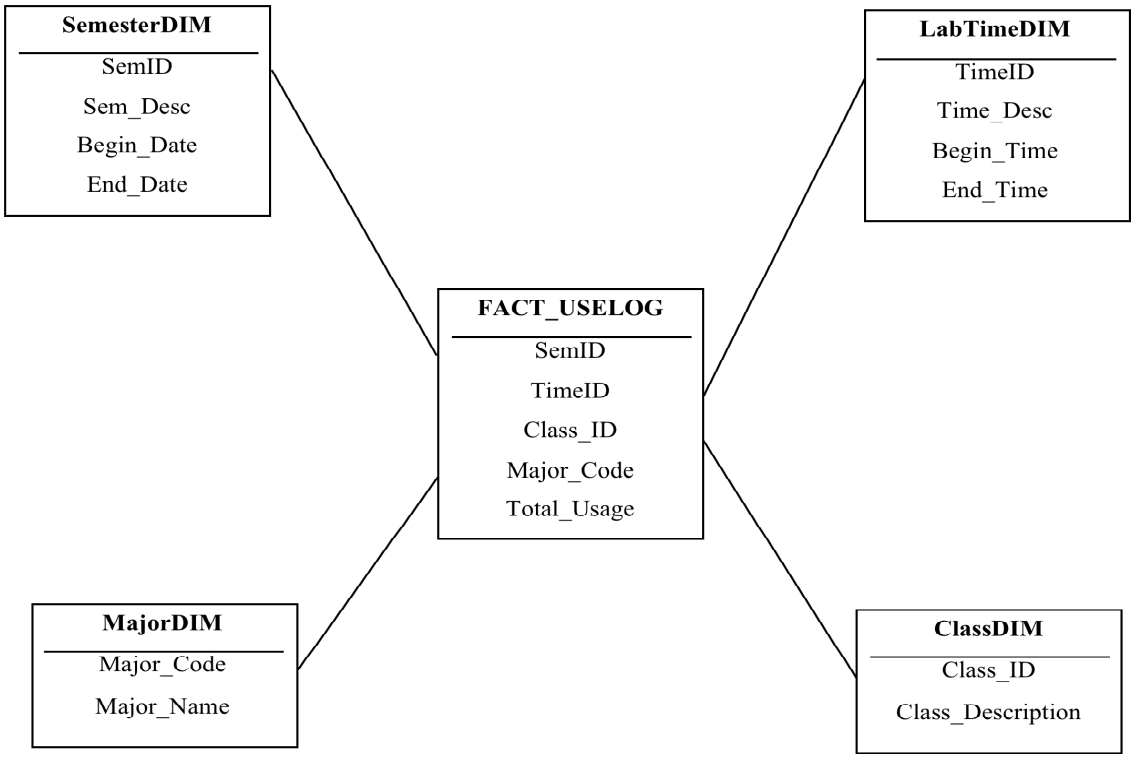
`dw.Major`: table that stores information about major codes and descriptions

`dw.Student`: table that stores information about students as described above

`dw.Uselog`: table that stores information about lab usage as described above

You do not need to copy these four tables (**`dw.Class`**, **`dw.Major`**, **`dw.Student`**, and **`dw.Uselog`**) into your account. You can just simply use these tables.

Solutions




```
--first create the dimensions
--create semester dimension
create table semesterDIM
(SemID      varchar2(10),
 Sem_Desc   varchar2(20),
 begin_date date,
 end_date   date);

-- create time dimension (note do not use time as a
-- table name, it is a reserve keyword)
create table labtimeDIM
(TimeID     number,
Time_Desc   varchar2(15),
begin_time  date,
end_time    date);

-- create major and class dimensions
create table majorDIM as
select * from dw.major;

create table classDIM as
select * from dw.class;
```

```
-- populate semester dimension
-- (the begin and end date can be changed)

insert into semesterDIM values ('S1', 'Semester1',
to_date('01-JAN', 'DD-MON'), to_date('15-JUL', 'DD-MON'));

insert into semesterDIM values ('S2', 'Semester2',
to_date('16-JUL', 'DD-MON'), to_date('31-DEC', 'DD-MON'));

--populate labtime dimension
insert into labtimeDIM values(1, 'morning', to_date('06:01',
'HH24:MI'), to_date('12:00', 'HH24:MI'));

insert into labtimeDIM values(2, 'afternoon', to_date('12:01',
'HH24:MI'), to_date('18:00', 'HH24:MI'));

insert into labtimeDIM values(3, 'night', to_date('18:01',
'HH24:MI'), to_date('06:00', 'HH24:MI'));


-- secondly, create a temp table to extract from uselog table
create table tempfact_uselog as
select U.log_date , U.log_time,
U.student_ID, S.class_id, S.major_code
from dw.uselog U, dw.student S
where U.student_id = S.student_id;

-- add a column in the tempfact table to store timeid
-- (cannot directly do this in the tempfact table because
-- log_time was of DATE type and timeid is of NUMBER type).

alter table tempfact_uselog
add (timeid number);

update tempfact_uselog
set timeid = 1
where  to_char(log_time, 'HH24:MI') >= '06:01'
and to_char(log_time, 'HH24:MI') <='12:00';

-- 62832 rows updated.


update tempfact_uselog
set timeid = 2
where  to_char(log_time, 'HH24:MI') >= '12:01'
and to_char(log_time, 'HH24:MI') <='18:00';

-- 76113 rows updated.
```

```
-- note that we use OR in the last update statement to
-- include the time between 18:01 and 06:00.

update tempfact_uselog
set timeid = 3
where to_char(log_time, 'HH24:MI') >= '18:01'
or to_char(log_time, 'HH24:MI') <='06:00';

-- 31665 rows updated.


-- alternatively, you may want to update timeid=3
-- for all other records where the time_id is still empty
      -- update tempfact_uselog
      -- set timeid = 3
      -- where timeid is NULL;



-- add a column in the tempfact_uselog table to store semid
-- (cannot directly do this in the test table because
-- log_date was of DATE type and semid is of VARCHAR type.)

alter table tempfact_uselog
add (semid varchar2(10));


-- populate the new attribute semid by summarizing
-- the date(log_date)

update tempfact_uselog
set semid = 'S1'
where to_char(log_date, 'MMDD') >= '0101'
and to_char(log_date, 'MMDD') <= '0715';

-- 91188 rows updated.


update tempfact_uselog
set semid = 'S2'
where to_char(log_date, 'MMDD') >= '0716'
and to_char(log_date, 'MMDD') <= '1231';

-- 79422 rows updated.
```

```
-- Now, create the fact table,
-- make sure to include the TOTAL aggregate.
-- This is an aggregate table of the earlier tempfact table.

create table fact_uselog as
select t.semid, t.timeid, t.class_id,
t.major_code, count(t.student_id) as total_usage
from tempfact_uselog t
group by t.semid, t.timeid, t.class_id, t.major_code;
```

Is everything ok? It looks ok, doesn't it?

Now, let's look at the data more carefully. If you don't look at the data, you will never know what has happened.

There are several questions that you need to ask yourself about the data:

1. How many records in the operational database?
2. How many records in the data warehouse?
3. What kind of data is in the operational database?
4. How do the tables look like in the data warehouse?

These questions will be a good start in **data exploration**.
Let's start with the operational database.

```
Select count(*) from dw.student;

       COUNT(*)
     ----------
          37951

Select count(*) from dw.uselog;

       COUNT(*)
     ----------
         108267

Select count(*) from dw.major;

       COUNT(*)
     ----------
            172

Select count(*) from dw.class;

       COUNT(*)
     ----------
             10
```

Now, let's do the same with the data warehouse. Tables classDIM and majorDIM are directly exported from dw.class and dw.major; so, we know them. Tables semesterDIM and labtimeDIM are manually created and we have inserted the records; so, we know them too.

Now, let's check tempfact_uselog and fact_uselog;

```
Select count(*) from tempfact_uselog;

        COUNT(*)
      ----------
          170610

Select count(*) from fact_uselog;

        COUNT(*)
      ----------
            1363
```

170610 records in tempfact_uselog have reduced to 1363 records in fact_uselog. This is due to the group by operation, where records having the same groups (i.e. same classes, same majors, same semesters, same labtimes) are grouped into a single record in the fact table, and an aggregate value (count of student_id) is also calculated. So, a reduction in number of records in fact_uselog is somehow expected.

Now, let's compare the data warehouse and the operational database. There are 170610 records in tempfact_uselog. Tempfact_uselog table is a join between tables dw.uselog and dw.student.

```
create table tempfact_uselog as
select U.log_date , U.log_time,
U.student_ID, S.class_id, S.major_code
from dw.uselog U, dw.student S
where U.student_id = S.student_id;
```

There are **108267** records in dw.uselog, and **37951** records in dw.student. The relationship between dw.student and dw.uselog is 1-many; meaning that one student may have used the lab more than one time in different time. Hence, a join between dw.student and dw.uselog is based on student_id in dw.student and student_id in dw.uselog.

Regardless how many records in dw.student, if dw.student and dw.uselog is 1-many (and no illegal students are allowed to use the lab), then the join between these two tables (which is basically table tempfact_uselog) would surely produce exactly the number of records in dw.uselog, which is **108267**.

BUT, table tempfact_uselog has **170610** records.

WHY is it different from dw.uselog?
Suspect something fishy???

So, this means that **tempfact_uselog** is **INCORRECT**. It has more records than expected. In fact tempfact_uselog has **62343 more records** than dw.uselog, which is not a small amount by any measure. Why does tempfact_uselog have these extra records? Can you see the mistakes from the codes above?
I can't, and nobody can.
The reason is that there is no mistake in the codes.

So, where is the mistake?
You need to look at the data. This is **data exploration** – to find what has happened, where the mistake was, etc. If you don't look at the data, you will not be able to see anything wrong, because the codes do not show any mistakes.

Because the number of records in dw.uselog and tempfact_uselog is big, you would not want to select all from the two tables to check visually; because it is just impossible (this is one of the reasons why I use big tables, so that you will not be able to check everything visually; rather, you need to use your logic).

Although it is difficult to check big tables visually, there is no harm to try.

```
Select * from dw.uselog;
Select * from tempfact_uselog;
```

Or rather, try these:

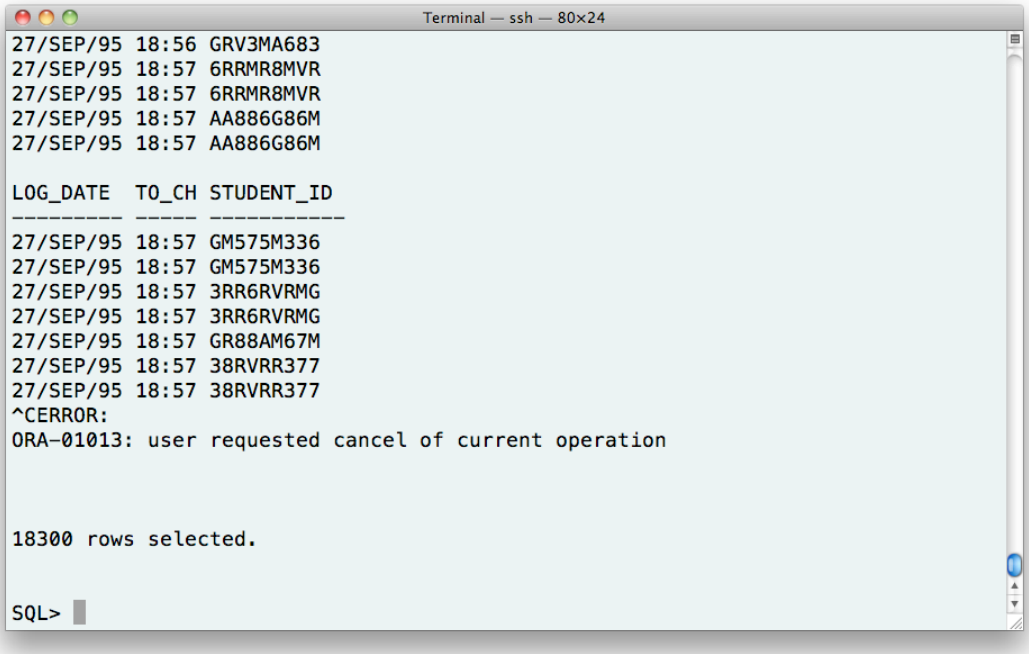
```
Select log_date, to_char(log_time, 'HH24:MI'), student_ID, act
From dw.uselog;
```

Or try the select all with some sort of ordering (Order By).

Now, check tempfact_uselog;

```
Select log_date, to_char(log_time, 'HH24:MI'), student_ID
From tempfact_uselog;
```

Did you notice anything unusual? This is a sample snapshot of the result.



It seems there are some duplicates (e.g. the same students use the lab at the same time). This is certainly wrong. How did it happen? Is there anything wrong with the code to create tempfact_uselog?

```
create table tempfact_uselog as
select U.log_date , U.log_time,
U.student_ID, S.class_id, S.major_code
from dw.uselog U, dw.student S
where U.student_id = S.student_id;
```

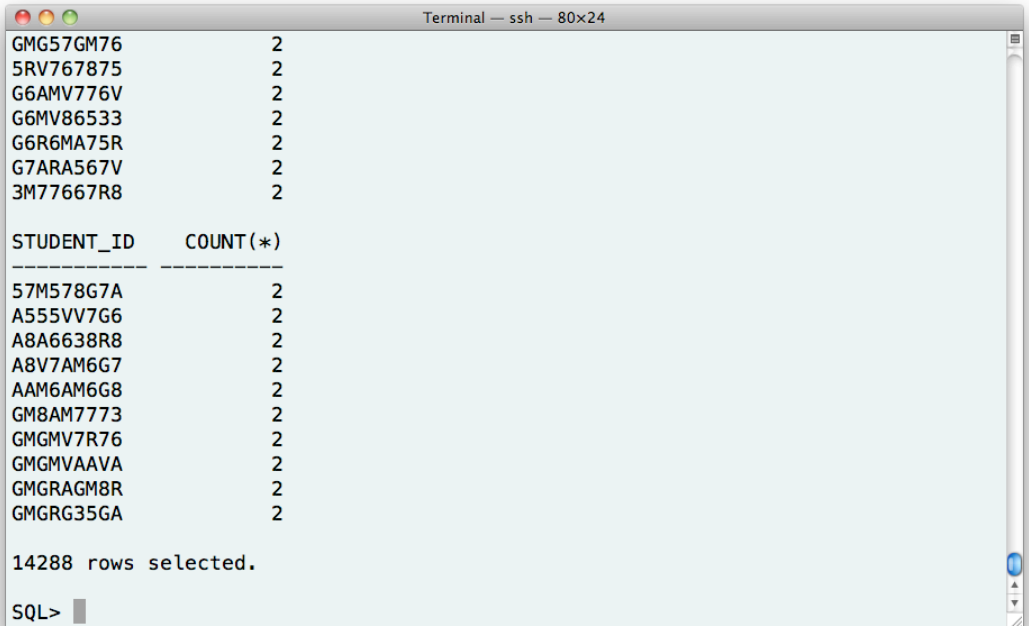
Or was it wrong in the alter and update commands to the tempfact_uselog table? No, it can't be, because alter and update commands do not add records. But create table tempfact_uselog is not wrong, either.

So, if the mistake is not with the code, could the mistake be from the input data itself? Let's explore the input data (e.g. dw.uselog and dw.student). Let's take one student as a sample. Let's choose "38RVRR377" (see the above screenshot. This student appears twice in the tempfact_uselog).

```
select *
from dw.student
where dw.student.student_id = '38RVRR377';

SE FU TYPE CLASS_ MAJOR_CO STUDENT_ID
-- -- ---- ------ -------- -----------
F  F  U    FR     BAd3     38RVRR377
F  F  U    FR     BAd3     38RVRR377
```

Student "38RVRR377" exists twice (duplicate) in dw.student. So, could there be other students who exists more than one in dw.student? Now, let's find it out.

```
select student_id, count(*)
from dw.student
group by student_id
having count(*) > 1;
```



There is not just one student duplicated in dw.student. There are 14288 students duplicated in dw.student.

The moral lesson is: "Don't trust the operational databases". They can be wrong. They can be incomplete. They can be inconsistent. There are many reasons why these have happened to the operational databases.

So, when you build a data warehouse, you need to clean the input data. This is called **Data Cleaning**.

There are many ways to perform data cleaning. Intuitively, in this case, we could copy the operational database (e.g. dw.student) to your local database, and clean the duplicate records.

Alternatively, we add DISTINCT when we reconstruct the tempfact table, so that the duplicate join results in tempfact will be eliminated by the DISTINCT clause in SQL.

```
create table tempfact_uselog2 as
select distinct U.log_date , U.log_time,
U.student_ID, S.class_id, S.major_code
from dw.uselog U, dw.student S
where U.student_id = S.student_id;

        COUNT(*)
    ----------
        108261
```

Now, there are only **108261 records** in tempfact_uselog2; recall that dw.uselog has **108267 records**. Tempfact_uselog2 is short of 6 records. It seems we have not totally solved the problem.

Maybe, there is something wrong too in dw.uselog. Remember, we should not trust the operational data. So, let's investigate further.

```
-- check if there are illegal students in dw.uselog
select * from dw.uselog
where student_id NOT IN
   (select student_id from dw.student);

-- no rows selected


-- check if there are illegal majors in dw.student
select *
from dw.uselog, dw.student
where dw.uselog.student_id = dw.student.student_id
and dw.student.major_code NOT IN
   (select major_code from dw.major);

-- no rows selected
```

```
-- check if there are invalid class in dw.student
select *
from dw.uselog, dw.student
where dw.uselog.student_id = dw.student.student_id
and dw.student.class_id NOT IN
   (select class_id from dw.class);

-- no rows selected


-- check if there are records in uselog not in tempfact_uselog
select *
from dw.uselog
where log_date NOT IN
   (select log_date from tempfact_uselog)
and log_time NOT IN
   (select log_time from tempfact_uselog)
and student_id NOT IN
   (select student_id from tempfact_uselog);

-- no rows selected


select
  to_char(log_time, 'HH24:MI') log_time,
  log_date,
  student_id,
  act,
  count(*)
from dw.uselog
group by log_time, log_date, student_id, act
having count(*) > 1;

      LOG_T LOG_DATE  STUDENT_ID  A   COUNT(*)
      ----- --------- ----------- - ----------
      20:14 03/APR/95 GM586MGA5   P          2
      15:54 08/MAR/97 GMR5M678M   P          2
      20:14 02/APR/96 GM586MGA5   P          2
      15:54 08/MAR/96 GMR5M678M   P          2
      15:54 09/MAR/95 GMR5M678M   P          2
      20:14 02/APR/97 GM586MGA5   P          2

      6 rows selected.
```

Bingo, even dw.uselog is incorrect. There are 6 duplicate records in dw.uselog. But, when DISTINCT is used the join between dw.uselog and dw.student, the duplicate problem is eliminated.

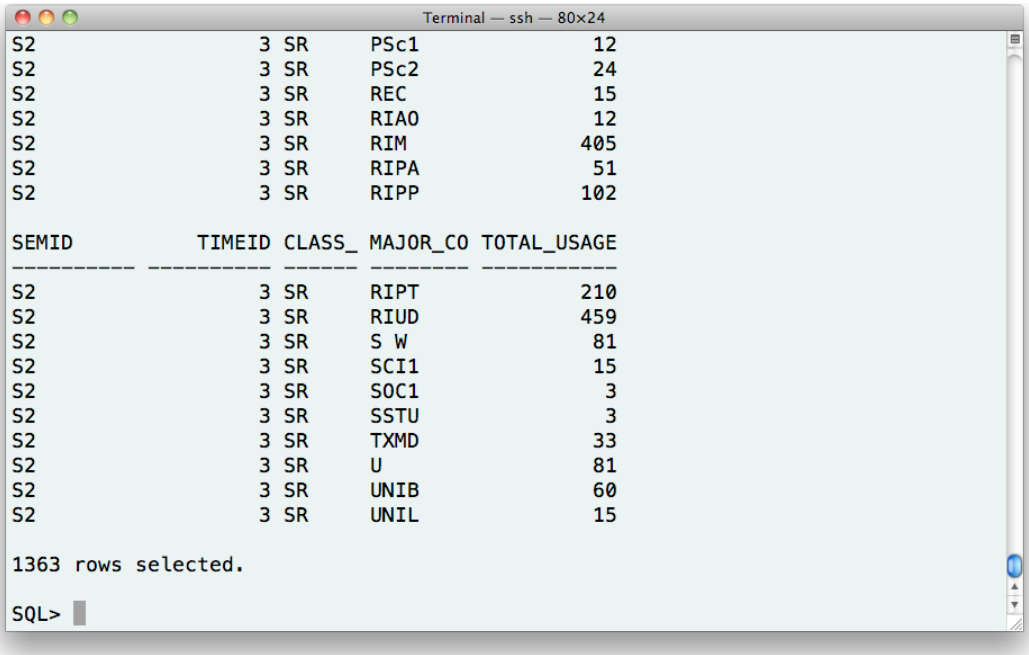
Hence, tempfact_uselog2 which has **108261 records** is in fact CORRECT, whereas dw.uselog having **108267 records** is not accurate.

Since tempfact_uselog2 is now correct, we can continue with the alter and update operations for the new tempfact_uselog2.

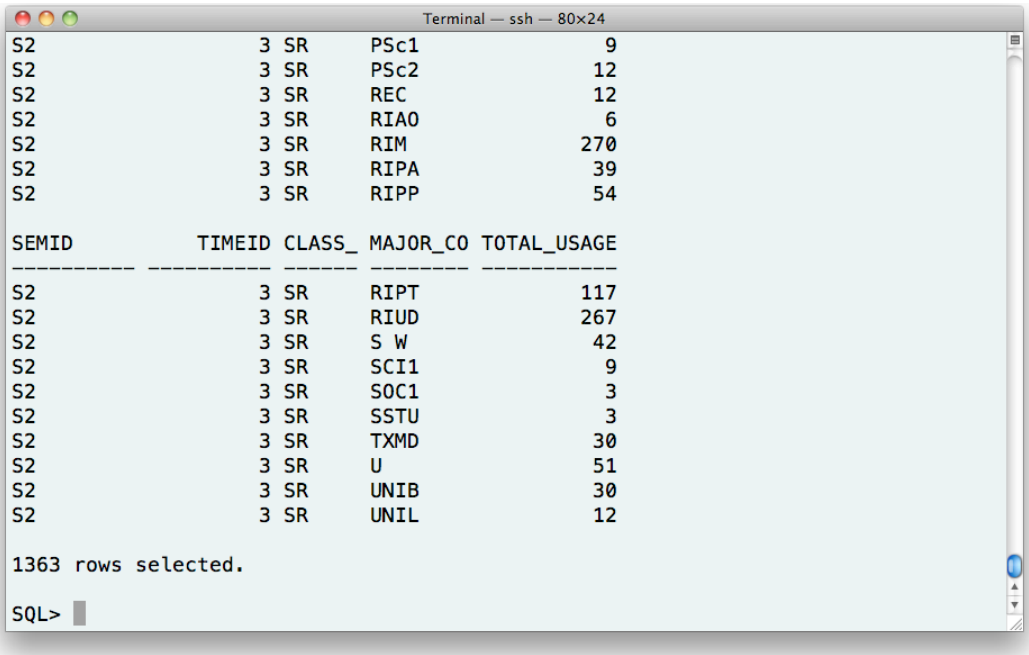
```
alter table tempfact_uselog2
add (timeid number);

update tempfact_uselog2
set timeid = 1
where  to_char(log_time, 'HH24:MI') >= '06:01'
and to_char(log_time, 'HH24:MI') <='12:00';

-- 39921 rows updated.


update tempfact_uselog2
set timeid = 2
where  to_char(log_time, 'HH24:MI') >= '12:01'
and to_char(log_time, 'HH24:MI') <='18:00';

-- 48261 rows updated.


update tempfact_uselog2
set timeid = 3
where to_char(log_time, 'HH24:MI') >= '18:01'
or to_char(log_time, 'HH24:MI') <='06:00';

-- 20079 rows updated.



alter table tempfact_uselog2
add (semid varchar2(10));

update tempfact_uselog2
set semid = 'S1'
where to_char(log_date, 'MMDD') >= '0101'
and to_char(log_date, 'MMDD') <= '0715';

-- 57612 rows updated.


update tempfact_uselog2
set semid = 'S2'
where to_char(log_date, 'MMDD') >= '0716'
and to_char(log_date, 'MMDD') <= '1231';

-- 50649 rows updated.



create table fact_uselog2 as
select t.semid, t.timeid, t.class_id,
t.major_code, count(t.student_id) as total_usage
from tempfact_uselog2 t
group by t.semid, t.timeid, t.class_id, t.major_code;
```

Now let's compare fact_uselog and fact_uselog2.

```
select *
from fact_uselog
order by semid, timeid, class_id, major_code;
```



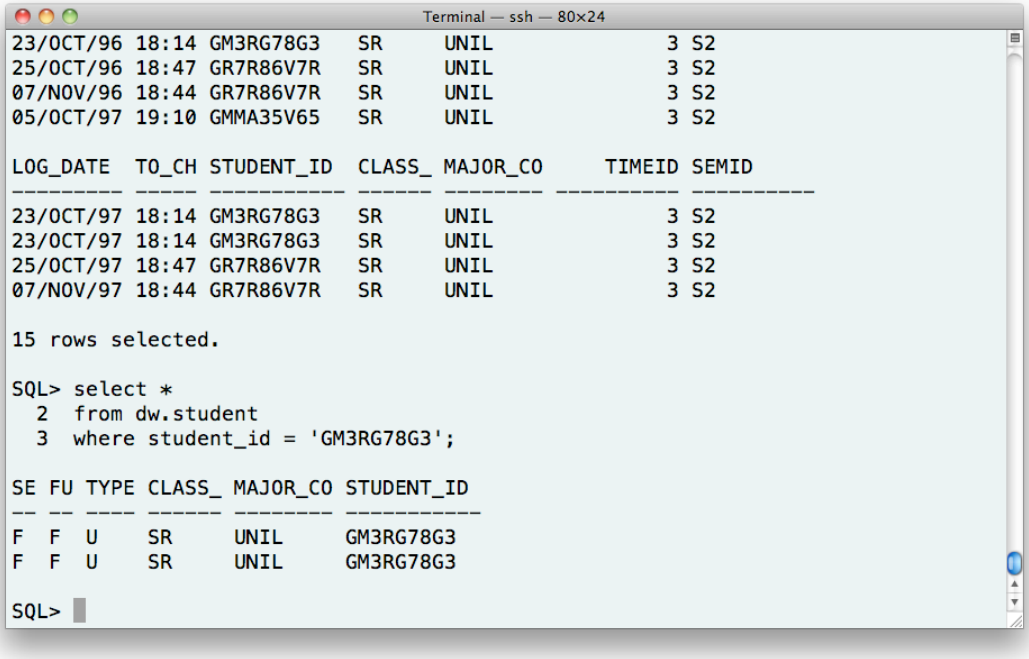

```
select *
from fact_uselog2
order by semid, timeid, class_id, major_code;
```



Although both fact_uselog and fact_uselog2 tables have the same number of records, their contents, especially total_usage, are different. This is because of the duplicate student records.

See a sample record below:

```
select log_date, to_char(log_time, 'HH24:MI'), student_id,
class_id, major_code, timeid, semid
from tempfact_uselog
where semid = 'S2'
and timeid=3 and major_code='UNIL'
order by log_date;
```



Now, let's continue by answering the three queries:

```
--query a) answer
select u.timeid, l.time_desc, sum(u.total_usage)
from fact_uselog2 u, labtimeDIM l
where u.timeid = l.timeid
group by u.timeid, l.time_desc;
```

```
TIMEID TIME_DESC       SUM(U.TOTAL_USAGE)
---------- --------------- ------------------
         3 night                     20079
         2 afternoon                 48261
         1 morning                   39921
```

```
--query b) answer
select u.timeid, u.major_code, u.class_id, sum(u.total_usage)
from fact_uselog2 u
group by u.timeid, u.major_code, u.class_id;

773 rows selected.


--alternative solution which gives more meaningful results
select t.time_desc, m.major_name, c.class_description, sum(u.total_usage)
from fact_uselog2 u, majorDIM m, classDIM c, labtimeDIM t
where u.major_code = m.major_code
and u.class_id = c.class_id
and u.timeid = t.timeid
group by t.time_desc, m.major_name, c.class_description;

722 rows selected. (WHY IS THIS LESS THAN 773 records?)
```

Hint:
Select * from MajorDIM
And notice that there are several majors having the same description. Hence, the alternative solution is not accurate.

```
--query c) answer
select u.major_code, u.semid, sum(u.total_usage)
from fact_uselog2 u
group by u.major_code, u.semid;

207 rows selected.
```

THE END