

SQL Tips

Finding Duplicates

```
select fields
from table
group by fields
having count(*) >1
```

Use this to find records which have the same values on the fields specified in the 'group by' statement. This will work only for tables which have a primary key.

For example, find students who have multiple records for the same unit and census date:

```
select person_id, course_cd, unit_cd, unit_census_dt, tr_org_unit_cd
from govt_student_load_liability
group by person_id, course_cd, unit_cd, unit_census_dt, tr_org_unit_cd
having count(*) >1
```

Note that the fields listed in the select statement must exist in the group by statement. Not all fields listed in the group by statement need to exist in the select statement however.

Finding The Maximum Record Count

```
select field, count(field)
from table
group by field
having count(field)=
(select max(a.cnt) from (select count(field) as cnt
from table
group by (field)) as a)
```

Use this to find the record which appears most often in a set of values e.g. the most often reported unit, the staff member with the most research grants.

Selecting All Just From Some Tables

```
select tablename1.fieldname, tablename2.*  
from tablename1, tablename2  
(join)
```

Use this when you want to return just one/few fields from one table but all the fields from another.

For example, to add student name to govt_student_enrolment table fields:

```
select person.surname, govt_student_enrolment.*  
from person, govt_student_enrolment
```

Selecting From A Restricted Set

```
select fields  
from table  
where conditional_field in  
(select conditional_field from table2)
```

Use this when you want to restrict your query to a certain set of records based on the output of a second query.

For example, to return student names only for students who are in submission 2:

```
select pe.surname  
from person pe  
where person_id in  
(select person_id from govt_student_enrolment  
where submission_yr = '2008' and submission_number = '2')
```

Handling Dates

Converting to a financial year:

```
select to_char(add_months(sysdate, -6), 'yyyy') || '/' ||  
to_char(add_months(sysdate, 6), 'yy')
```

Converting from character format to date format:

```
select to_date('200608', 'yyyymm')
```

Converting from date format to character format:

```
select to_char(sysdate, 'yyyymm')
```

Selecting the current month:

```
select trunc(sysdate, 'month')
```

Using Decode

```
decode(value, search_value, result, default_value)
```

Use to more efficiently perform IF-THEN-ELSE type clauses. It will compare the value with the search_value and return the result if true, the default_value if false (default_value is optional).

For example, to group students by arrival year:

```
select person_id,  
decode(arrival_yr, 0000, 'never arrived', 0001, 'born in australia', 'arrived in australia')  
arrival_grp  
from govt_student_enrolment
```

Using Case

Similar to decode but easier to write and read and you can use boolean logic. Only available in Oracle 9i and above.

```
CASE
  WHEN quantity > 20 THEN 0.75
  WHEN quantity BETWEEN 10 AND 20 THEN 0.90
  ELSE 0.95
END;
```

For example, to apply a variable weighting to load based on course level:

```
select course_level,
case course_level
  when 'level 1' then eftsI *3
  when 'level 2' then eftsI *2
  else eftsI
end
"weighted load",
from student_course_level
```