SQL notes and things.  
  
  
Here are some things that ill likely forget:

Log in to server:  
  
 nki38@csora131 - welcome <- login credentials for terminal sql.

To use SQL Developer from our lab machines:

- Launch SQL Developer (from the Programming menu).

- Select the + button (top left) to add a new connection.

- Fill out the properties box with:

Connection name: csora131

Username: your uni usercode

Password: your Oracle password (initially welcome).

- For the "Oracle" tab underneath:

Connection Type: TNS

Network Alias: CSORA131

Creating a table:

Basic:

CREATE TABLE <NAME>

(first varchar(4) not null,

Second smallint,

Age int check (Age between 0 and 100),

Constraint check (first = first or first = first) <- Allows this

PRIMARY\_KEY (first, second));

Foreign Key:

CREATE TABLE new\_table  
(Item varchar(6) not null,

Something integer not null,

ref\_guy integer not null references ref\_guy, <- Points for join condition in sql

Check\_guy varchar(12) not null

Check (Role in (‘tuple1’,’tuple2’......)),

Primary key (Something, Item)

Foreign key (something, else) references MATCH);

As the primary key for MATCH is The foreign key for new\_table here.

Inserting into the tables:

INSERT INTO TABLE (COL1, Col2, OL3, etc) Values (23,12,312,312,3);

INSERT INTO *table\_name* (*column1*, *column2*, *column3*, ...)

VALUES (*value1*, *value2*, *value3*, ...);

SQL QUERIES!!!!

HAVING:

THE GREATEST VALUE OF ALL SOMETHING ELSE)

ALSO WE CAN USE WHERE SEEDNO = MAX(subquerry)

Select name, count(\*) as something

From somethingelse

Where an attribute Not null

Group by name

Having count(\*) > all(select count(\*)

From player

Where something else is not null

Groupby name));

We can also use:

ANY:

To compare across relations

MINUS:

((Select distinct something from table)

Minus

(Select distinct something from table where (true thing)):  
**SUBSTITUTION**

Not really but:

Select something, hi, A.name, B.Name

From player A, Player B, something

Where (true) and (subquery)

Otherwise

select Draw, No  
from match  
where (select distinct country from player where id=player1) =   
 (select distinct country from player where id = player2);

UPDATE

Update table

Set duration = 3,

Meh = 4,

Sheh = 4

Winner = (select id from subquery…..)

Where draw = ‘final’

^ The whole thing is compiled before execution , just a wee note

([UPDATE TABLE] -> [SET VALUES] -> [WHERE CONDITION])

Where can have names a nd all of that jazz, ya know.

-VIEWS-

Views are quick look thingies

Create VIew as View\_name

As select id,name,count(\*) as somethinelse

From this,that

Where (conditions)

Group by Id,name;

Viewing Views:

Select id ,name

From View\_name

Where somethinelse =4 ;

-TRIGGERS-

Create or replace trigger TRIGGER\_NAME

Before insert on judging

For each row

When (new.role in(‘referee’,’char umpire’))

Declare

roleExists integer;

Begin

Select count(\*) into RoleExists

From judging

Where daw = :new.draw and match =: new.match and role =:new.role;

If roleExists = 1 then

Raise\_application\_error (num => -20055)

Msg (“can only be one chair umpire or ref per match)

End if;

End;

CREATE [OR REPLACE ] TRIGGER trigger\_name   
{BEFORE | AFTER | INSTEAD OF }   
{INSERT [OR] | UPDATE [OR] | DELETE}   
[OF col\_name]   
ON table\_name   
[REFERENCING OLD AS o NEW AS n]   
[FOR EACH ROW]   
WHEN (condition)   
DECLARE   
 Declaration-statements   
BEGIN   
 Executable-statements   
EXCEPTION   
 Exception-handling-statements   
END;

Triggers are used both for not breaking shit and for some pretty outputs

CREATE OR REPLACE TRIGGER display\_salary\_changes   
BEFORE DELETE OR INSERT OR UPDATE ON customers   
FOR EACH ROW   
WHEN (NEW.ID > 0)   
DECLARE   
 sal\_diff number;   
BEGIN   
 sal\_diff := :NEW.salary - :OLD.salary;   
 dbms\_output.put\_line('Old salary: ' || :OLD.salary);   
 dbms\_output.put\_line('New salary: ' || :NEW.salary);   
 dbms\_output.put\_line('Salary difference: ' || sal\_diff);   
END;   
/

-DELETE-

DELETE FROM TABLE  
WHERE SHIT IS TRUE

This is a join on relation -> relation -> relation using double joins

SELECT DISTINCT C\_Name

FROM Creature C

JOIN Achievement A ON (C.C\_ID = A.C\_ID)

JOIN Skill S ON (A.S\_Code = S.S\_Code)

WHERE S.S\_Weight <= 0.5

ORDER BY C\_Name;

This same relation can be done as a nested loope,

JOINS:

Inner Join - Everything in both

( \_ ( The crossover) \_)

Left Outer

( \* ( \* ) \_ ) - Join on a parameter and use null values for table values not provided by 2

Right Outer

( \_ ( \* ) \* ) - Join the other way around here

Groupby:  
Prolly wanna make an excel table or something

But we groupby att1,then att2, then att3 then att 4 having

SUMMARY THINGS  
SELECT (CustomerID),  
Country  
FROM Customers  
GROUP BY COUNTRY  
ORDER BY COUNT(CUSTOMERID) DESC;