CS4600 Database Theory and Application – Basic SQL Commands

Dr. Kaoning Hu

This table is for your reference. However, this table is not enough to describe all the commands that you need to operate a database. You need to study the details by yourselves.

Command	Example
create table	**create table** *student*(*ID* **char**(9) **primary key**, *name* **varchar**(20) **not null**, *dept_name* **varchar**(20), *tot_cred* **numeric**(4,1), **foreign key** *(dept_name*) **references** *department)*);
drop table	**drop table** *r*;
alter table	**alter table** *r* **add** *A D*; **alter table** *r* **drop** *A*;
insert into	**insert into** *student* **values** ('3003', 'Green', 'Finance', *null*);
delete from	**delete from** *instructor* **where** *dept_name*= 'Finance';
update	**update** *instructor* **set** *salary* = *salary** 1.03 **where** *salary* > 100000;
case	**update** *instructor* **set** *salary* = **case** **when** *salary* <= 100000 **then** *salary* * 1.05 **else** *salary* * 1.03 **end**;
select from where (You may choose to keep or remove duplicated tuples by parameter "**all**" or "**distinct**")	**select** *name* **from** *instructor* **where** *dept_name='*Comp. Sci.' **and** *salary* > 80000; **select** ∗ **from** *instructor, teaches*; (Cartesian product) **select** * **from** *instructor* **natural join** *teaches*; (natural join)
"nested subqueries"	**select distinct** *course_id* **from** *section* **where** *semester* = 'Fall' **and** *year*= 2009 **and**

	course_id **in** (**select** *course_id* **from** *section* **where** *semester* = 'Spring' and *year*= 2010);
as	**select** *ID, name, salary/12* **as** *monthly_salary* **from** *instructor*;
"string operations"	**select** *name* **from** *instructor* **where** *name* **like** '%dar%';
"logic operations" and, or, not	Be careful with comparisons with "**null**".
"set operations" union intersect except	**select … from … where …** **union** **select … from … where …** You can only perform set operations when the two relations have the same attributes. Duplicates can be kept if you choose to use the parameter "**all**" after set operation.
"aggregate functions" avg, min, max, sum, count All aggregate operations except count(*) ignore null values.	**select avg**(*salary*) **from** *instructor* **where** *dept_name*= 'Comp. Sci.'; You can only perform one aggregate function at a time.
group by	**select** *dept_name*, **avg**(*salary*) **from** *instructor* **group by** *dept_name*;
having *vs.* where Predicates in the **having** clause are applied after the formation of groups whereas predicates in the **where** clause are applied before forming groups.	**select** *dept_name*, **avg**(*salary*) **from** *instructor* **group by** *dept_name* **having avg**(*salary*) > 42000;
"set comparison" >, <, =, some, all,…	**select** *name* **from** *instructor* **where** *salary* > **some** (**select** *salary* **from** *instructor* **where** *dept name* = 'Biology');
"test empty relations"	**select … from … where exists …** **select … from … where not exists…**
"test duplicates"	**select … from … where unique…**
"derived relations"	**select** *dept_name*, *avg_salary* **from** (**select** *dept_name*, **avg**(*salary*) **as** *avg_salary* **from** *instructor*

	group by *dept_name*) **where** *avg_salary*> 42000;
with (temporary view of relation)	**with** (**select max**(*budget*) **from** *department*) **as** *maximun_budget*(*value*) **select** *budget* **from** *department, maximum_budget* **where** *department.budget*= *maximum_budget.value*;
"scalar subquery"	**select** *dept_name*, (**select count**(*) **from** *instructor* **where** *department.dept_name*= *instructor.dept_name*) **as** *num_instructors* **from** *department*;