Database Notes

SELECT STATEMENTS

```
select upper(substr(ename, 2, 1)) || lower(substr(ename, 2)) from emp;

select ename, sal from emp order by sal;

select job, avg(sal) from emp group by job;

select * from emp;

select from emp where sal < 800 or sal > 1500

select ename from emp where upper(job) = 'MANAGER';

select ename, dname from emp e, dept d where e.deptno = d.deptno;
        e and d are given names for tables so col. in the table with the same name can
        be distinguished
        e and d can also be used before they are declared

select ename, sal from emp where ename like '_A%E' order by ename;
```

Note: Like is a form of regular expression

```
select ename, empno, job from emp where sal between 1000 and 3000;

select ename from emp where comm is not null;

select distinct job form emp;
```

Note: ** is a block comment

```
select ename from emp, customer where emp.deptno = customer.repid;

select ename, sal + if null(comm, 0);
```

Types of join: join, natural join, cross join, inner join, left/right outer join

Note: Natural join joins the col with the same name

```
Sqlite3: select date('now');
       select time('now');
```

Note: An inner select can see tables from an outer select

```
select name, count(ordid) from customer natural join ord group by name order by name;
```

List the customers name and their representatives name for all customers in California.

sqlite> select name, ename from emp, customer where empno = repid AND lower(state) = 'ca';

List the empno and ename for all emp who are clerks or analysts and who do not earn btw 850 and 1200.

sqlite> select ename, empno from emp where job in('ANALYST','CLERK') and sal between 850 and 1200;

List the name of employees who manage others and the number of people they manage. Not all employees who manage have the job title manager.

sqlite> select m.ename, count(m.ename) from emp e, emp m where e.mgr = m.empno group by e.mgr;

Create a select to list product sales. Specifically list the repid and name, customer id and name, product id and description, and the total dollar amount of sales of the product to the company, when the amounts are > 1000 and the companies are in CA.

sqlite> select c.repid, e.ename, c.custid, c.name, p.prodid, p.descrip, i.itemtot from emp e natural join customer c natural join ord d natural join product p natural join item i where e.empno in(c.repid) and i.itemtot > 1000 and c.state = 'CA' limit 5;

Using IN with a sub select find the names of all employees wh oare not customer representatives.

sqlite> select ename from emp where empno not in(select repid from customer);

Using EXISTS with a sub select find the names of all employees who are not customer rep.

sqlite> select ename from emp where not exists(select repid from customer where repid = empno);

List the avg sal by job title where the avg sal is > 2500. order by desc order of avg sal.

sqlite> select job, avg(sal) from emp group by job having avg(sal) > 2500 order by avg(sal) desc;

List all dept by name and the number of employees of each when they have more than five employees.

```
sqlite> select dname, count(ename) from dept d, emp e where e.deptno = d.deptno
group by dname
```

Find the total for each order where the total is computed by multiplying the ACTUALPRICE and QTY of each item in the order.

```
sqlite> select d.ordid, actualprice * qty AS total from ord d natural join item i group by
d.ordid;
```

```
select ename, decode (trim(job), 'CLERK', 'a clerk',
                          'SALESMAN', 'a salesperson',
                          'ANALYST', 'an analyst'
                          '', 'xxx') as "Job", sal from emp;

select ename,
   case trim(job)
      when 'CLERK' then 'A clerk'
      when 'SALESMAN' then 'A salesperson'
      else 'MANAGEMENT'
   end, sal from emp;

--Examples of select statements involving a table with a nested table
select BreederName, N.Name, N.Birthdate from Breeders, TABLE(Breeders.Animals) N;

select BreederName, N.Name, N.Birthdate from Breeders, TABLE(Breeders.Animals) N
where N.Name = 'Tank';
);
```

How to select tables of a database in oracle:

```
Select table_name, owner From user_tables order by owner, table_name
```

CREATE, UPDATE, INSERT, AND DELETE STATEMENTS

```
update emp set job = 'MANAGER', deptno = 20, sal = sal + 1000 where ename =
'JONES';

insert into emp2 (empno, ename, job, mgr, sal, deptno) values (7799, 'TAYLOR',
'COMPSCI', 7779, 2900, 10);

delete from emp2 where job = 'MANAGER';

Create view em(empname, Mname) as select e.ename, m.ename from emp e, emp m
where e.mgr = m.empno;
```

```
create table emp2 as select * from emp;

create or replace type name_ty as object(
last varchar2(20),
first varchar2(20),
mid varchar2(20)
)

insert all
   into junk value(1, sysdate)
   .
   .
   .
   into junk value(100, sysdate)
select * from dual;
create global temporary table name(
   col type,
   .
   .
   .
) on commit delete rows

--Nested tables: tables that are stored in a table (sortof)
--_Ty is just a convention, but a good one to follow

--An object with the name Animal_ty
create type Animal_ty as object(
   breed       varchar2(25),
   name        varhcar2(25),
   birthdate   date);
/

--Creating a nested table called Animals_NT from Animal_Ty
create type Animals_NT as table of Animal_Ty;
/

--Creates a table Breeders with variables name and nested table of animals
create table Breeders(
   BreederName     varchar2(25),
   Animals         Animals_NT)
nested table Animals store as Animals_NT_TAB;

--s at the end of animals is important for convention

--How to insert into a table that has a nested table
insert into Breeders(BreederName, Animals) values('Jane Doe',
```

```
Animals_NT(
Animal_Ty('cat','dickens','15-apr-00'),
Animal_Ty('dog','Dog','15-sep-07'),
Animal_Ty('dog','Tank','14-sep-09')
)
```

--Example of why to use a nested table: if keeping records of employee there may be dependents which can be held in a nested table

--Examples of using the insert, update, and delete statments with a table that has a nested table

```
insert into TABLE(select Animals from Breeders where BreederName = 'Jane Doe')
values (Animal_ty('dog','charlie','01-Jan-15'));

update TABLE(select Animals from Breeders where BreederName = 'Jane Doe') N set
N.Birthdate = '25-Dec-10' where N.Name = 'charlie';

delete TABLE(select Animals from Breeders where BreederName = 'Jane Doe') N
where N.Name = 'Tank';
```

```
SQL> create table emp3(
  2  id      number(5),
  3  last    varchar2(15),
  4  first   varchar2(12),
  5  mgr     number(5),
  6  name    varchar(30) generated always as(initcap(last) || ', ' || initcap(fi
rst)) virtual,
  7  constraint emp3_pk primary key(id),
  8  constraint emp3_fk foreign key(mgr) references emp3(id)
  9  )
 10  /
```

DATE STATEMENTS

Show how to convert the date 14 October 1066 into the form used by oracle

```
SQL> select TO_DATE('14 October 1066') FROM DUAL;
```

Show how to convert the date June 25, 745 B.C. into Oracle form.

```
SQL> select TO_DATE('June 25, 745 B.C.', 'mon dd, yyyy B.C.') from DUAL;
```

Convert your birth date into the number of days from Jan 1, 4712 B.C.

```
SQL> select to_date('jan 1, 1995', 'mon dd, yyyy') - to_date(1, 'j') + 1 from dual;
```

Show how to convert the oracle date 23-FEB-12 to the form 23 Februray, 2012

```
SQL> select to_char(to_date('23-FEB-12', 'dd-mon-yy'), 'dd month, yyyy') from dual;
```

EXTRACT (YEAR) FROM date time expression

YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, TIMEZONE_HOUR,
TIME_ZONE_MINUTE, TIMEZONE_REGION

DATE SYSDATE, CURRENT_DATE

SYSDATE -> date on oracle server
CURRENT_DATE -> might be diff. if in another time zone

TIMESTAMP(d)
  d can be 0-9
    0 -> secound, 1 -> tenth of second, ..., 9 -> billionth of second

REGULAR EXPRESSION

REGULAR EXPRESSION / REGEXP
  REGEXP_SUBSTR
  REGEXP_INSTR
  REGEXP_LIKE
  REGEXP_REPLACE
  REGEXP_COUNT

REGEXP Notation
  [abc] -> matches an a, b, or c
  * -> matches zero or more
  + -> matches one or more
  ^ -> if first char: beginning, after: means not
  $ -> end
  .* -> any # of anything

```
select regexp_substr('123-456-789', '-[^-]+') from dual;

select ename from emp where regexp_like(ename, '^[^aeiou]+', 'i');
```

i -> case insensitive
e -> case sensitive

PROCEDURES, TRIGGERS, AND FUNCTIONS

```
--Procedure for entering a row with min. amt. of values to emp
create or replace procedure new_emp(id in number, name in char, d in number)
    is
    begin
        insert into emp(empno, ename, deptno) values(id, name, d);
    end;
/

--How to execute the procedure
execute new_emp(33,'Tay',10);

--Creating a procedure to enter min. amt. of values to dept
create or replace procedure new_dept(id in number, name in char)
    is
        number_exception exception;
    begin
        if id < 0 or id > 99 then
            raise number_exception;
        end if;
        insert into dept(deptno, dname) values(id, name);

       exception
           when number_exception then
               raise_application_error(-20101,'Invalid Department Number');
   end;
/

--How to execute the procedure
execute new_dept(55, 'NewDept')


--Trigger for updating names to uppercase when inserted or updated
create or replace trigger emp_bef_upd_ins_row2
    before insert or update of ename on emp
    for each row
    begin
        :new.ename := upper(:new.ename);
    end;
/

SQL> create or replace function square(n number) return number
  2  is
  3  begin
  4      return n*n;
  5  end square;
  6  /
```

```
SQL> create or replace function factr(n natural) return number
 2  is
 3      y number := 1;
 4  begin
 5      if n > 1 then
 6          y := n * factr(n-1);
 7      end if;
 8      return y;
 9  end factr;
10  /

SQL> create or replace function today
 2  return varchar2
 3  as
 4  begin
 5      return to_char(sysdate,'MM/DD/YYYY');
 6  end;
 7  /

SQL> create or replace function myDate(d in date)
 2      return varchar2
 3      is
 4          strDate varchar2(25);
 5      begin
 6          strDate := to_char(d, 'fmMonth dd, yyyy');
 7          return(strDate);
 8      end;
 9  /
```

THEORY OF DATABASE

A primary key uniquely identifies a row in a table.
Can't have more than one in the same table

DKNF -> Domain Key Normal Form

Relation -> theoretical table that exists mathematically
Table -> 2 dim. array with headers

Under first normal form, all occurrences of a record type must contain the same number of fields.

1st normal form:
Trying to get the table to be atomic. No primary key should be shown more than once.

Solution: breaking the original table into smaller tables, that when joined together you get the original table with no duplicate data, more than one primary key, and no data with more than one value.

Under second and third normal forms, a non-key field must provide a fact about the key, us the whole key, and nothing but the key. In addition, the record must satisfy first normal form.

2nd normal form:
Must be in first normal form. Deals with a partial dependency.

dependency - columns that depend on the primary key.

partial dependency - A dependency that is dependent on two primary keys from different tables.

Example:
You have an author table , book-author table, and book table. All the info about the author is in the author table and everything about the book is in the

Second normal form is violated when a non-key field is a fact about a subset of a key.

Third normal form is violated when a non-key field is a fact about another non-key field

Fourth [5] and fifth [6] normal forms deal with multi-valued facts. The multi-valued fact may correspond to a many-to-many relationship, as with employees and skills, or to a many-to-one relationship, as with the children of an employee (assuming only one parent is an employee). By "many-to-many" we mean that an employee may have several skills, and a skill may belong to several employees.

Note that we look at the many-to-one relationship between children and fathers as a single-valued fact about a child but a multi-valued fact about a father.

In a sense, fourth and fifth normal forms are also about composite keys. These normal forms attempt to minimize the number of fields involved in a composite key.

Under fourth normal form, a record type should not contain two or more independent multi-valued facts about an entity. In addition, the record must satisfy third normal form.

fourth normal form is defined in terms of multivalued dependencies, which correspond to our independent multi-valued facts

Fifth normal form deals with cases where information can be reconstructed from smaller pieces of information that can be maintained with less redundancy. Second, third, and

fourth normal forms also serve this purpose, but fifth normal form generalizes to cases not covered by the others.

Address(street, city, state, zip)
  zip -> state so street, city, zip and zip, state
  Leave in original form, b/c to inefficient in normal form

R(A, B, C) (= R(A,B) * R(A,C), there might be additional rows on right
* means join on common col.

4th normal form requires R(A,B,C) = R(A,B) * R(A,C)

Multivalue Dependency: R(A,B,C) != R(A,B) * R(A,C)

Join dependency -> decompse into more than 2 pieces, which follow a constraint

loseless decomposition -> when the decomposed tables are joined together to get the original table

Try to avoid boolean values

Deletion Anomally -> have something valid and something is deleted to make it not valid.

Insertion Anomally -> inserting a new row and table becomes invalid

Update Anomally -> updating table make it invalid

```
--Table named SUPPLY that holds a progect, part, and name
create table SUPPLY(
Sname varchar2(25),
Part  varchar2(25),
Project varchar2(10))
/

--EXAMPLE OF JOIN DEPENDENCY

--Inserting data into the table SUPPLY
insert into SUPPLY (Sname,Part,Project) values('Smith','Bolt','X');
insert into SUPPLY (Sname,Part,Project) values('Smith','Nut','Y');
insert into SUPPLY (Sname,Part,Project) values('Adamsky','Bolt','Y');
insert into SUPPLY (Sname,Part,Project) values('Walton','Nut','Z');
insert into SUPPLY (Sname,Part,Project) values('Adamsky','Nail','X');

--Tables created out of parts of the SUPPLY table
create table Tab1 as select Sname, Part from SUPPLY;
```

```
create table Tab2 as select Sname, Project from SUPPLY;
create table Tab3 as select Part, Project from SUPPLY;

--Selecting the tables and joining them together
select distinct * from Tab1 natural join Tab2 natural join Tab3;

--Output from the select statement
Smith|Bolt|X
Smith|Bolt|Y
Smith|Nut|Y
Adamsky|Bolt|X
Adamsky|Bolt|Y
Walton|Nut|Z
Adamsky|Nail|X

/*It can be seen that the table made from joining the tables, that are pieces of
the original table, do not produce the original table. The new table has more rows
than the original, which violates R(A,B,C) = R[A,B] * R[A,C].

--END OF JOIN DEPENDENCY EXAMPLE
```

OTHER

ASC - ascending
DESC - descending

Oracle commands:
password - allows you to change the password
describe - gives the info/schema of the table
Show all - gives list of settings

How to set up oracle database in shortcut:
E:\CS440\instantclient_11_2\sqlplus.exe
cs44003/Taytay14@oracle.cs.semo.edu:1521/orcl

Types:
- Integer
- Real
- Numeric
- text
- blob

-- is a comment

Operands: <, >, <=, >=, !=, NULL, is NULL, ||
Functions: trim, abs, upper, lower, length, ltrim, rtrim, Between, And, Or, Not, Order by, Like, Distinct or Unique

STEPS TO MAKE DATABASE:

1. change mode to csv
2. import the csv file
3. save a database
4. open the database

Data Dictionary

USER_CATALOG (CAT)
USER_OBJECTS (OBJ)
USER_TABS (TABS)
USER_VIEWS
USER_CONSTRAINTS

DICT tablename, comment
TAB, COL

USER_, ALL_, DBA_
USER_TABLES
USER_COL_COMMENTS
USER_INDEXES
USER_COMMENTS
USER_TRIGGERS
USER_VIEWS
USER_USERS

SOUNDEX -> takes name and produces a letter and 3 digits

DATA TYPES IN ORACLE:

NUMBER(5,2), 5 numbers with 2 decimal pts
CHAR
DATE
TIMESTAMP
VARCHAR
LONG
RAW
LONGRAW
BLOB
CLOB
BFILE

CSV -> comma seperated value

Alter table statement can be dangerous

Godel: Discovered any mathematical sys. that is complicated has statements that are true or false but are not provable.

USER_SEQUENCE

create sequence seqname
    optional: increment by, default: 1
        start with, default: 1
        max value, default: no max
        min value, default: no min
        cycle, default: no

Alter sequence
Drop sequence

Ada -> Procedures/Funcitons
    Procedure does not return value
    Functions return values

block -> denoted by same kind of braces
trigger -> operates in background, makes sure things stary the same.

USER_TAB_COMMENTS
USER_COL_COMMENTS

SQL PLUS
    Column <col><opt1><opt2>...
    format A<n> (alphanumeric)
    format 99,999 (numbers)
    heading <text>
    null <text>
    --All of these commands only deal with how things look

--Using the sql developer
Steps:
Open sql developer
click on green plus sign
enter username and password
Change Hostname to oracle.cs.semo.edu
Change to service name and enter orcl