Gradebook Database

drop table catalog cascade constraints;
create table catalog (  
cno varchar2(7),
title varchar2(50),
primary key (cno));
drop table courses cascade constraints;
create table courses (  
  term        varchar2(10),  
  lineno      number(4),  
  cno         varchar2(7) not null,  
  a           number(2) check(a > 0),  
  b           number(2) check(b > 0),  
  c           number(2) check(c > 0),  
  d           number(2) check(d > 0),  
  primary key (term,lineno),  
  foreign key (cno) references catalog);
drop table components cascade constraints;
create table components (  
term varchar2(10),
lineno number(4) check(lineno >= 1000),
compname varchar2(15),
maxpoints number(4) not null check(maxpoints >= 0),
weight number(2) check(weight>=0),
primary key (term,lineno,compname),
foreign key (term,lineno) references courses);
drop table students cascade constraints;
create table students ( 
sid varchar2(5),
fname varchar2(20),
lname varchar2(20) not null,
minit char,
primary key (sid));

drop table enrolls cascade constraints;
create table enrolls ( 
sid varchar2(5),
term varchar2(10),
lineno number(4),
primary key (sid,term,lineno),
foreign key (sid) references students,
foreign key (term,lineno) references courses);
drop table scores cascade constraints;
create table scores (
    sid      varchar2(5),
    term     varchar2(10),
    lineno   number(4),
    compname varchar2(15),
    points   number(4) check(points >= 0),
    primary key (sid,term,lineno,compname),
    foreign key (sid,term,lineno) references enrolls,
    foreign key (term,lineno,compname) 
        references components);
Mailorder Database

drop table zipcodes cascade constraints;
create table zipcodes (  
   zip number(5) primary key,  
   city varchar2(30));

drop table employees cascade constraints;
create table employees (  
   eno number(4) primary key,  
   ename varchar2(30),  
   zip number(5) references zipcodes,  
   hdate date);
drop table parts cascade constraints;
create table parts(
    pno number(5) primary key,
    pname varchar2(30),
    qoh integer check(qoh >= 0),
    price number(6,2) check(price >= 0.0),
    olevel integer);

drop table customers cascade constraints;
create table customers ( 
    cno number(5) primary key,
    cname varchar2(30),
    street varchar2(30),
    zip number(5) references zipcodes,
    phone char(12));
drop table orders cascade constraints;
create table orders (  
one number(5) primary key,  
cno number(5) references customers,  
eno number(4) references employees,  
received date,  
shipped date);  

drop table odetails cascade constraints;
create table odetails (  
one number(5) references orders,  
pno number(5) references parts,  
qty integer check(qty > 0),  
primary key (ono,pno));
Alter Table

alter table customers add (  
    fax  char(12),  
    ctype char check(ctype in ('I','B'))  
);  

alter table customers modify (  
    street   varchar2(50)  
);
SQL: Insert Statement

insert into components values
(’f96’,1031,’exam1’,100,30);
insert into courses values
(’f96’,1031,’csc226’,90,80,65,50);
insert into courses(term,lineno,cno) values
(’f96’,1037,’csc326’);
insert into enrolls(term,lineno,sid) values
(’f96’,1031,’1111’);
SQL: Querying the Database

1. Get `pno` and `pname` values of parts that are priced less than $20.00.

   ```sql
   select pno,pname
   from   parts
   where  price < 20.00;
   ```

2. Get all the rows of the employees table.

   ```sql
   select *
   from   employees;
   ```

3. Get `pno` values for parts for which orders have been placed. Eliminate duplicate answers.

   ```sql
   select distinct pno
   from   odetails;
   ```
4. Get all details of customers whose names begin with the letter ”A”.

   select *
   from customers
   where cname like 'A%';

5. Get the orderno and cname values for customers whose orders have not yet been shipped (i.e. the shipped column has a null value).

   select orderno,cname
   from orders,customers
   where customers.cno = orders.cno and
       shipped is null;

6. Get sid values of students who have scores between 50 and 70 points in any component of any course they have enrolled in.

   select sid
   from scores
   where points between 50 and 70;
7. Get `cname` and `ename` pairs such that the customer with name `cname` has placed an order through the employee with name `ename`.

   ```sql
   select distinct cname, ename
   from customers, orders, employees
   where customers.cno = orders.cno and
     employees.eno = orders.eno;
   ```

8. For each `odetail` row, get `ono`, `pno`, `pname`, `qty`, `price` values along with the total price for this item. The total price is simply the product of unit price and quantity.

   ```sql
   select x.ono, x.pno, p.pname, x.qty,
     p.price, (x.qty * p.price) total
   from odetails x, parts p
   where x.pno = p.pno
   ```
9. Get all pairs of \texttt{cno} values for customers based in the same zipcode.

\begin{verbatim}
select c1.cno, c2.cno
from customers c1, customers c2
where c1.zip = c2.zip and c1.cno < c2.cno;
\end{verbatim}

10. Get \texttt{pno} values for parts that have been ordered by at least two different customers.

\begin{verbatim}
select distinct y1.pno
from orders x1, orders x2,
     odetails y1, odetails y2
where y1.pno = y2.pno and
     y1.ono = x1.ono and
     y2.ono = x2.ono and
     x1.cno < x2.cno
\end{verbatim}
11. Get `c_name` values of customers who place orders with employees living in the Fort Dodge.

```sql
select distinct c_name
from orders,customers
where orders.cno = customers.cno and
eno in (select eno
    from employees,zipcodes
    where employees.zip = zipcodes.zip
    and city = 'Fort Dodge');
```

12. Get `c_name` values of customers living in Fort Dodge or Liberal.

```sql
select c_name
from customers,zipcodes
where customers.zip = zipcodes.zip and
city in ('Fort Dodge','Liberal');
```
13. Get **pname** values for parts with the least price.

```sql
select pname
from parts
where price <= all (select price
                      from parts);
```

14. Get the **pname** values of parts that cost less than the least priced **Land Before Time** part.

```sql
select pname
from parts
where price < all
  (select price
   from parts
   where pname like 'Land Before Time%');
```
15. Get \texttt{cname} values of customers who have placed at least one order through employee with \texttt{eno} = 1000.

\begin{verbatim}
select cname
from customers
where exists (select 'a'
               from orders
               where orders.cno = customers.cno and
                     eno = 1000);
\end{verbatim}

16. Get \texttt{cname} values of customers who do not place any orders through employee with \texttt{eno} = 1000.

\begin{verbatim}
select cname
from customers
where not exists
  (select 'a'
   from orders x
   where orders.cno = customers.cno and eno = 1000);
\end{verbatim}
17. Get cno values of customers who have placed an order for both parts, pno = 10506 and pno = 10507, in the same order.

```sql
select cno
from orders
where exists (select 'a'
              from odetails
              where odetails.ono = orders.ono and
              odetails.pno = 10506) and
exists (select 'a'
        from odetails
        where odetails.ono = orders.ono and
        odetails.pno = 10507);
```
18. Get cities in which customers or employees are located.

```sql
select city
from customers, zipcodes
where customers.zip = zipcodes.zip
union
select city
from employees, zipcodes
where employees.zip = zipcodes.zip
```
19. Get \texttt{cno} values of customers who place orders with ALL employees from \textit{Wichita}.

\begin{verbatim}
select c.cno
from customers c
where not exists
  (select *
   from employees e
   where e.city = 'Wichita' and
      not exists (select *
                      from orders x
                      where x.cno = c.cno and
                          x.eno = e.eno));
\end{verbatim}
20. Get total quantity of part 10601 that has been ordered.

```sql
select sum(qty) TOTAL
from   odetails
where  pno = 10601;
```

21. Get the total sales in dollars on all orders.

```sql
select sum(price*qty) TOTAL_SALES
from   orders,odetails,parts
where  orders.ono = odetails.ono and
       odetails.pno = parts.pno;
```

22. Get the number of cities in which customers are based.

```sql
select count(distinct city)
from   customers, zipcodes
where  customers.zip = zipcodes.zip;
```
23. Get the \texttt{pname} values of parts that cost more than the average cost of all parts.

\begin{verbatim}
select pname
from parts
where price > (select avg(price)
    from parts);
\end{verbatim}

24. For each part, get \texttt{pno} and \texttt{pname} values along with total sales in dollars.

\begin{verbatim}
select parts.pno,pname,sum(qty*price) TOTAL_SALES
from orders,odetails,parts
where orders.ono = odetails.ono and
    odetails.pno = parts.pno
group by parts.pno,pname;
\end{verbatim}
25. Get employee name, employee number, part name, part number, together with total quantity each employee supplies of that part to customers with cno values 1111 or 2222.

```sql
select e.eno, e.name, p.ono, p.name, sum(qty)
from orders x, parts p, employees e, odets od
where x.ono = od.ono and x.eno = e.eno and
      od.pno = p.pno and x.cno in (1111, 2222)
group by e.eno, e.name, p.pno, p.name;
```

26. For each part, get pno and pname values along with total sales in dollars, only when the total sales exceeds 1000 dollars.

```sql
select parts.pno, parts.name, sum(qty*price) TOTAL_SALES
from orders, odets, parts
where orders.ono = odets.ono and
      odets.pno = parts.pno
group by parts.pno, parts.name
having sum(qty*price) > 1000;
```
27. Get **pno** and **pname** values of parts ordered by at least two different customers.

```sql
select parts.pno, parts.pname
from orders, odetails, parts
where orders.ono = odetails.ono and
    odetails.pno = parts.pno
group by parts.pno, parts.pname
having count(distinct cno) >= 2;
```
create view employee_sales as
    select employees.eno,ename,sum(price*qty) SALES
from employees,orders,odetails,pars
where employees.eno = orders.eno and
    orders.ono = odetails.ono and
    odetails.pno = parts.pno
group by employees.eno,ename;
SQL: Insert, Delete, Update

insert into cheap_parts
    select *
    from   parts
    where  price  <=  20.00;

insert into soso_parts
    select *
    from   parts
    where  price between 20.00 and 50.00;

insert into expensive_parts
    select *
    from   parts
    where  price  >  50.00;
• The `update` statement

  ```sql
  update parts
  set    qoh = qoh + 100
  where  qoh < 5*olevel;
  ```

  increases by 100 the `qoh` values of those rows of the `parts` table that have a `qoh` value less than 5 times the `olevel` value.

• The `update` statement

  ```sql
  update parts
  set    qoh = (select max(qoh)
                from   parts)
          where  qoh < 100;
  ```

  sets the `qoh` value of those parts whose current `qoh` value is less than 100 to the maximum `qoh` value present in the table. Notice the use of a select statement as an expression in the `set` clause.
• The update statement

```sql
update parts
set qoh = 2*qoh
where 3 <= (select sum(qty)
    from odetails
    where odetails.pno = parts.pno);
```

doubles the qoh values of those parts which have been ordered in quantities of 3 or more. Notice the sub-select in the where clause.
• The **delete** statement

```sql
delete from customers;
```
deletes all rows in the **customers** table.

• The **delete** statement

```sql
delete from customers
where zip in (select zip
from zipcodes
where city = 'Fort Hays');
```
deletes all customers who live in **Fort Hays**.
• The **`delete`** statement

```sql
delete from employees
where eno in (select eno
    from orders,odetails,parts
    where orders.ono = odetails.ono and
          odetails.pno = parts.pno
    group by eno
    having sum(price*qty) < 200);
```

deletes all employees who have total orders less than $200. Notice the sub-select statement in the where clause.
Sequences

create sequence <seq-name>
    [INCREMENT BY integer]
    [START WITH integer]
    [MAXVALUE integer | NOMAXVALUE]
    [MINVALUE integer | NOMINVALUE]
    [CYCLE|NOCYCLE]

create sequence custseq start with 1000;

insert into customers
    values(custseq.nextval, ’Jones’, ’123 Main St.’, 67226, ’111-111-1111’);
Oracle Data Dictionary

Some useful data dictionary tables:

dictionary(table_name, comments) ; public alias: dict

user_catalog(table_name, table_type) ; public alias: cat

user_objects(object_name, object_id, object_type, created,
             last_ddl_time, timestamp, status)

user_tables(table_name, tablespace_name, ...) ; public alias: tabs

user_tab_columns(table_name, column_name, data_type,
                  data_length, data_precision, data_scale, nullable, ...) ; public alias: cols

user_views(view_name, text_length, text)