1)
```
create table t2(
a1 number(10),
a2 number(10));
```
在建表的时候表名如果用括号括上，则表名是区分大小写！！！
```
begin
for i in 1..10 loop
insert into t2(a1,a2) values(i,i);
end loop;
commit;
end;
/
```

2)
```
declare
i number(10) := 1;
begin loop
insert into t3 values(i,i);
i:= i+1;
if i>10 then
exit;
end if;
end loop;
commit;
end;
/
```

3)
```
declare
i number(10):= 1;
begin
while i<=10 loop
 insert into t3 values(i,i);
i:=i+1;
end loop;
commit;
end;
/
```

4)演示游标如何使用。
```
create table t4 as select * from t3 where 1=2;
select count(*) from t4;

create or replace procedure proc1
as
v1 t3.a1%type;
v2 t3.a2%type;
cursor cur1 is select a1,a2 from t3;
begin
open cur1;
loop
fetch cur1 into v1,v2;
exit when cur1%notfound;
insert into t4(a1,a2) values(v1,v2);
end loop;
close cur1;
commit;
end;
/
```
存储过程创建完成后没有执行，需要手工执行。

```
execute proc1
或者
begin
proc1;
end;
/
```
这两种方法都是可以的。

5）演示异常处理

```
 create table t5(
a1 number(10) primary key,
a2 number(10));

create or replace procedure proc2
(v1 t5.a1%type,v2 t5.a2%type)              //这行是参数。
as
p1 t5.a1%type                               //这行是变量。
begin
select a1 into p1 from t5 where a1=v1;
exception
when no_data_found then
insert into t5(a1,a2) values(v1,v2);
commit;
end;
/

show error    //这个命令可以告诉哪行错了。
```

执行这个存储过程要用到参数，

```
execute proc2(1,11);
```

6)演示存储过程中通过参数把返回值返回。

```
create or replace procedure proc3(v1 out number)
as
select count(*) into v1 from t3;
end;
/
```

在 sqlplus 中调用

```
variable p1 number     //这里的变量不让定义长度。
execute proc3(:p1);
```

在存储过程中调用

```
declare
p1 number(10);
begin
proc3(p1);
dbms_output.put_line(p1);
end;
/
```

7)函数的例子。

```
create or replace function func1
return number is
v1 number(10);
begin
select count(*) into v1 from t3;
return v1;
```

```
end;
/
```
函数只能返回一个值，如果要返回多个，需要用到参数。
```
select func1 from dual;

declare
p1 number(10);
begin
p1:=func1;
dbms_output.put_line(p1);
end;
/
```

8)触发器的例子。
行触发器。
触发器有两个时机，在操作之前，和操作之后触发。

8.1 insert 触发器。
```
select * from t5;

create table t6 as select * from t5;    //此时 t5 和 t6 完全一样。

create or replace trigger tri1
before insert on t5
for each row
begin
insert into t6(a1,a2) values(:new.a1,:new.a2);
```
//new 是一个临时的，虚的表。这里会把 t5 中表的内容临时存放在 new 表中，然后再由 new 表转到 t6.new 表是在插数据的时候使用。
```
//old 表是 delete 时用到。
//update 用到了 new 和 old 表。
end;
/
```

触发器中不能写提交或者其他的事物语句，为了 ROLLBACK 考虑。

8.2delete 触发器。

```
create or replace trigger tri2
before delete on t5
for each row
begin
delete from t6 where a1=:old.a1 and a2=:old.a2;
end;
/
```

8.3 update 触发器。
```
create or repleace trigger tri3
before update on t5
for each row
degin
update t6 set a1=:new.a1,a2=:new.a2
where a1=:old.a1 and a2=:old.a2;
end;
/
```

9)包的例子。
包中包含存储过程和函数。

```
create or replace package pack1
```

```
as
procedure proc4(p1 t5.a1%type,p2 t5.a2%type);
procedure proc5(v1 out number);
end;
/


create or replace package body pack1
as
procedure proc4(p1 t5.a1%type,p2 t5.a2%type)
as
begin
insert into t5 values(p1,p2);
commit;
end;
procedure proc5(v1 out number)
as
begin
select count(*) into v1 from t5;
end;
end;
/

execute pack1.proc4(3,33)
```

10）最后介绍的视图。

```
desc user_source


col name format a10
col text format a50
select name,text from user_source;
```

11）常用的两个视图，在调优的时候能够用到。

```
SELECT CONSTRAINT_NAME,CONSTRAINT_TYPE,TABLE_NAME FROM USER_CONSTRAINTS;
```

查看那个表有什么类型的约束。

```
SELECT TABLE_NAME,INDEX_NAME FROM USER_INDEXES;
```

产看哪个表有哪个索引。

```
SELECT TABLE_NAME,INDEX_NAME,COLUMN_NAME FROM USER_IND_COLUMNS;
```

查找哪个索引在哪个列上。

12）约束写在表的最后用 constraint 的约束是表级约束，卸载列的后面是列级约束。

13）唯一约束是不能有重复的值。但是可以为空。在一个表中可以为每个列建立唯一约束。一般如果没有特殊要求，这个唯一约束是不建议建的，因为在操作的时候会严重影响性能。
而主键是不允许为空。并且一个表中只能有一个主键。

建主键和建唯一约束自动建立所以，这个自动建立的索引是不能手工删除的。只能删除主键和唯一约束。

14）参照完整性约束

```
SQL> CREATE TABLE DEPT10(
  2  DEPTNO NUMBER(2) NOT NULL,
  3  DEPTNAME VARCHAR2(40) NOT NULL,
  4  CONSTRAINT DEPT_PK PRIMARY KEY(DEPTNO));

Table created.

SQL> CREATE TABLE EMP10(
  2  EMPNO NUMBER(6) NOT NULL,
  3  EMPNAME VARCHAR2(40) NOT NULL,
  4  DEPTNO NUMBER(2),
```

```
  5  CONSTRAINT EMP_PK PRIMARY KEY(EMPNO),
  6  CONSTRAINT EMP_FK FOREIGN KEY(DEPTNO) REFERENCES DEPT10(DEPTNO));

Table created.
```

外键可以为空。

```
SQL> CREATE TABLE EMP10(
  2  EMPNO NUMBER(6) NOT NULL,
  3  EMPNAME VARCHAR2(40) NOT NULL,
  4  DEPTNO NUMBER(2),
  5  CONSTRAINT EMP_PK PRIMARY KEY(EMPNO),
  6  CONSTRAINT EMP_FK FOREIGN KEY(DEPTNO) REFERENCES DEPT10(DEPTNO) ON DELETE
CASCADE);
```

删除主键后，级联删除外键表中的数据。

15）检查 CHECK 约束。

1.不可为空即使检查约束。

16）新建表的方法

```
CREATE TABLE T3 AS SELECT * FROM T1;
```

建表并且插入数据。

```
CREATE TABLE T4 AS SELECT * FROM T1 WHERE 1=2;
INSERT INTO T4 SELECT * FROM T1;
```

建表和插入数据分别操作。

17）删除表

如果先删主键表，应该先删主外键约束。

```
DROP TABLE DEPT CASCADE CONSTRAINTS;
```

18)数据库链接

1.分布式数据库。在一个环境中有很多数据库。

为 s c o t t 用户授予创建数据库链接的权限。

```
grant create database link to scott;
```

在 s c o t t 用户中创建数据库链接

```
SQL> create database link hhh
  2  connect to scott identified by tiger
  3  using 'ora20';
```

s c o t t 用户操作某个节点上的表。

```
select * from tab@hhh
```

19)查看当前数据库可以打开的游标的个数

```
SQL> show parameter open

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
open_cursors                         integer     300
open_links                           integer     4
open_links_per_instance              integer     4
read_only_open_delayed               boolean     FALSE
session_max_open_files               integer     10
```

20）

在 s q l p l u s 中的的定义编辑器:

```
define_editor=gedit
```

输入 s q l 语句

输入 ed 命令打开编辑器编辑语句，保存退出。

再用/执行保存的语句。

21）记录我的输入内容

```
SQL> spool off
SQL> spool /home/oracle/haha.sql
SQL> select * from scott.emp;
SQL> spool off
```

22）查看已经创建的存储过程

```
desc user_source
select name,text from user_source
```

23)

```
alter trigger tri1 disable
alter trigger tri1 enable;
alter table t36 disable all triggers;
alter table t36 enable all triggers;
```