

SQL Assignment 80

INDEX

Exp.No.	Title Of Experiment	Page No.
1	Creation of tables	1
2	Insertion of values in to tables	7
3	Selection Queries	12
4	Updation Queries	22
5	Deletion Queries	23
6	Craetion of Views	24
7	PL/SQL Queries	27
8	Cursor	39
9	Trigger	41
10	Forms	74
11	Reports	90

CREATIONS OF TABLES

1.Sailors Table

SQL>create table sailors(sid number(5),sname varchar(20),rating number(5),age number(3));

O/P: Table Created.

```
SQL>desc sailors;
      sid number(5)
      sname varchar2(20)
      rating      number(5)
      age number(5)
```

SQL>alter table sailors add constraint c,primary key (sid);

O/P: 1 Table Created.

SQL> alter table sailors add constraints c2 check(age>17);

O/P: 1 Table Created.

SQL>desc sailors

```
      sid      not null  number(5)
      sname                varchar2(20)
      rating              number(5)
      age                 number(5)
```

2. Reserves Table

SQL>create table Reserves(bid number(5),sid number(5),day date, foreign key(bid) referncing boats,foreign key(sid) referencing sailors);

Table Created.

```
SQL>desc Reserves
      bid number(5)
      sid number(5)
      day date
```

3. Boats Table

```
SQL>create table boats(bid number(5),bname varchar2(20),bcolor
varchar2(20));
```

Table Created

```
SQL>desc boats
      bid number(5)
      bname varchar2(20)
      bcolor varchar(20)
```

```
SQL>alter table boats add constraints c3 primary key(bid);
```

```
SQL>Table Altered.
```

```
SQL>desc boats
      bid number(5)
      bname varchar2(20)
      bcolor varchar2(20)
```

4.Branch Table

```
SQL>create table branch(branch_name varchar2(20),branch_city
varchar2(20),assets number(20));
```

Table Created.

```
SQL>desc branch
      branch_name varchar2(20)
      branch_city varchar2(20)
      assests number(20)
```

5.Customer Table

```
SQL>create table customer(customer_name  
varchar2(20),customer_street varchar2(20),customer_city  
varchar2(20));
```

Table Created.

```
SQL>desc customer  
customer_name varchar2(20)  
customer_street varchar2(20)  
customer_city varchar2(20)
```

6.Loan Table

```
SQL>create table loan(branch_name varchar2(20),loan_number  
number(20),amount number(20));
```

Table Created.

```
SQL>alter table loan add constraint c1 foreign  
key(loan_number)referencing borrower(loan_number));
```

SQL>Table Altered

```
SQL>desc loan  
branch_name varchar2(20)  
loan_number number(20)  
amount number(20)
```

7.Borrower Table

```
SQL>create table borrower(customer_name  
varchar2(20),loan_number number(20));
```

Table Created.

```
SQL>desc borrower  
customer_name varchar2(20)  
loan_number number(20)
```

```
SQL>alter table borrower add constraint c2  
primarykey(loan_number);
```

SQL>Table Altered.

```
SQL>desc borrower  
customer_name varchar2(20)  
loan_number not null number(10)
```

8.Account Table

```
SQL>create table account(branch_name  
varchar2(20),account_number number(20),balance  
number(20),foreign key(account_number) referencing depositor);
```

Table Created.

```
SQL>desc account  
branch_name varchar2(20)  
account_number number(20)  
balance number(20)
```

9. Depositor Table

```
SQL>create table depositor(customer_name  
varchar2(20),account_number number(20));
```

Table Created.

```
SQL>desc depositor  
      customer_name varchar2(20)  
      account_number number(20)
```

```
SQL>alter table depositor add constraint c3 primary  
key(account_number);
```

SQL>Table Altered.

```
SQL>desc depositor  
      Customer_name not null varchar2(20)  
      Account_number          number(20)
```

```
SQL>alter table account add constraint c4 check(balance>1000);
```

SQL>Table Altered.

```
SQL>alter table loan add constraint c5 check(amount <1000000);
```

SQL>Table Altered.

INSERTION OF TABLES

SAILORS

SQL>insert into sailors values(&sid,&sname,&rating,&age);

SQL>enter sid:3

enter sname :pavan

enter rating:5

enter age:18

SQL>1 row created.

SQL>select * from sailors;

sid	sname	rating	age
3	pavani	5	18
4	rajii	4	20
5	padmaja	6	19
6	rohit	5	18

RESERVES

SQL> insert into reserves values(&bid,&sid,&day');

SQL>enter bid:4

enter sid:6

enter day:17-jul-07

SQL>1 row created.

SQL>select * from reserves;

bid	sid	day
4	6	17-jul-07
3	6	18-jul-07
2	7	19-jul-07
5	9	20-jul-07

BOATS

SQL>insert into boats values(&bid,&bname,&bcolor);

SQL>enter bid:4
enter bname:pavan
enter bcolor:blue

SQL>1 row created.

SQL>select * from boats;

bid	bname	bcolor
4	pavani	blue
6	raji	red
2	padmaja	green

BRANCH

SQL>insert into branch
values('&branch_name','&branch_city',&assets);

SQL>enter branch_name:prd
enter branch_city:ong
enter assets:50

SQL>select * from branch;

branch_name	branch_city	assets
prd	ong	50
erd	nel	51
gun	hyd	52
crd	ban	53
sus	vij	54

CUSTOMER

```
SQL>insert into customers  
values('&customer_name','&customer_street','&customer_city');
```

```
SQL>enter customer_name:pav  
      enter customer_street:suj  
      enter customer_city:ong
```

```
SQL>1 row created.
```

```
SQL>select * from customers;
```

customer_name	customer_street	customer_city
pav	suj	ong
ruh	nir	nel
neeru	sam	hyd
rajii	raj	ban
paddu	madur	vij

LOAN

```
SQL>insert into loan values('&branch_name','&loan_no,&amount);
```

```
SQL>enter branch_name:prd  
      enter loan_no:501  
      enter amount:50
```

```
SQL>1 row created.
```

```
SQL>select * from loan;
```

branch_name	loan_no	amount
prd	501	50
crd	502	51
gun	503	52
crd	504	53

Borrower

SQL>insert into borrower values('&customer_name',&loan_no);

SQL>enter customer_name:pav
enter loan_no:501

SQL>1 row created.

SQL>select * from borrower;

customer_name	loan_no
pav	501
roh	502
raji	503
paddu	504
neeru	505

ACCOUNT

SQL>insert into account
values('&branch_name',&account_no,&balance);

SQL>enter branch_name:prd
enter account_no:60
enter balance:500

SQL>1 row created.

SQL>select * from account;

branch_name	account_no	balance
prd	60	500
erd	61	550
gun	62	600
crd	63	700
subj	64	800

DEPOSITOR

SQL>insert into depositor values('&customer_name',&account_no);

SQL>enter customer_name:pav

Enter account_no:60

SQL>1 row created

SQL>select * from depositor;

customer_name	account_no
pav	60
roh	61
rajii	62
paddu	63
neeru	64

SELECTION QUERIES

1. Find the names of all branches in the loan relation?

SQL>select l.branch_name from loan l;

O/P::: branch_name
 prd
 erd
 gun
 crd
 suj

2. Find all loan numbers for loans made at the 'prd' branch with loan amount>49?

SQL>select l.loan_number from loan l where l.amount>49 and
l.branch_name='prd';

O/P::: loan_number
 501

3. Find the loan numbers of these loans with loan amount>=50 and <=54?

SQL>select l.loan_number from loan l where l.amount>=49 and
l.amount<=54;

O/P::: loan_number
 501
 502
 503
 504
 505

4. Find all customers who have a loan on the bank display their names and loan numbers?

SQL>select b.customer_name,b.loan_number from borrower b,loan l where b.loan_number=l.loan_number;

O/P:::	customer_name	loan_number
	pav	501
	ruh	502
	raji	503
	paddu	504
	neeru	505

5. Find the names and loan numbers of all the customers who have a loan at 'prd' branch?

SQL>select B.customer_name,l.loan_number from borrower B,loan l where B.loan_number=l.loan_number and l.branch_name='prd';

O/P::: pav 501

6. Find the names of all branches that have assets greater than at least one branch located in ong.

SQL>select T.branch_name from branch T,branch S where T.assets>S.assets and S.branch_city='ong';

O/P::: erd
gun
crd
suj
erd
gun
erd
suj
erd

7. Find all the customers whose name starts with 'p'?

SQL>select c.customer_name from customer c where
c.customer_name like 'p%';

O/P::: pav
paddu

8. Find the names of all customers whose street address includes the substring 'pmainv'?

SQL>select customer_name from customer c where
c.customer_name like 'pmainv';

O/P::: no rows selected.

9..Find the ages of sailors whose names begins and ends with 'B' has atleast 3 characters?

SQL>select s.sailor_age from sailors s where s.sailor_name like 'B-
%B';

O/P:::no rows selected.

10.Complete increaments fro the ratings of person who have sailed to different boats on the same date?

SQL>select s.rating+1 from sailors s,reserves r1,reserves r2 where
r.sid>r2.day and r1.day=r2.day and r1.sid=s.sid and s.sid=r2.sid;

O/P::: no rows selected.

11. The select branch_name and loan_number and increment the amount value 100 times from the loan relation?

SQL>select branch_name,loan_number,amount*100 from loan;

O/P::: branch_name loan_number Amount*100

prd	501	5000
erd	502	5100
gun	503	5200
crd	504	5300
suji	505	5400

12. List the entire loan relation in descending order of amount if several loans have the same amount order them in ascending order by loan number?

SQL>select * from loan order by amount desc,loan_number asc;

O/P::: branch_name loan_number amount

suji	505	54
crd	505	53
gun	503	52
erd	502	51
prd	501	50

13. Find all the customers names having loan or account or both?

SQL>select customer_name from borrower union
select customer_name from depositor;

O/P:::customer_name

neeru
paddu
pav
raji
roh

14. Find all the customers who have an account but no loan at the bank?

SQL>select customer_name from depositor EXCEPT select customer_name from borrower;

O/P::: no rows selected.

15. Display the minimum rating value from sailors table?

SQL>select MIN(s.rating)from sailors s;

O/P::: min(s.rating)
4

16. Find the average account varies at prd branch?

SQL>select avg(balance) from account where branch_name='prd';

O/P::: avg(balance)
525

17. Find the average account balance for each branch?

SQL> select branch_name,AVG(balance) from account group by branch_name;

branch_name	AVG(balance)
erd	735
erd	578
gun	630
prd	525
suj	840

18. Find the number of depositors for each branch?

SQL>select a.branch_name,count(distinct d.customer_name) from depositor d,account a where a.account_number=d.account_number group by a.branch_name;

O/P:::branch_name	count(Distincted customer_name)
crd	1
erd	1
gun	1
prd	1
subj	1

19.Find the average of balance for each branch, average of balance of branch must be greater than 700?

SQL>select branch_name,avg(balance) from account group by branch_name having avg(balance)>700;

O/P::: branch_name	avg(balance)
crd	891
subj	1019

20.Find all customers who have both a loan and an account at the bank?

SQL>select customer_name from borrower where customer_name in(select customer_name from depositor);

O/P::: customer_name
neeru
paddu
pav
rajii
roh

21. Find all the customers who have both a loan and account at prd branch?

```
SQL> select distinct customer_name from borrower b, loan l where  
b.loan_number=l.loan_number and branch_name='prd'  
and (branch_name, customer_name) in (select  
branch_name, customer_name from depositor d, account a where  
d.account_no=a.account_no);
```

```
O/P::: customer_name  
pav
```

22. Find all the customers who have a loan at bank but do not have an account at the bank?

```
SQL> select customer_name from borrower where  
customer_name in (select customer_name from depositor);
```

```
O/P::: no rows selected.
```

23. Find the names of the all branches that have assets greater than those of at least one branch located in ong?

```
SQL> select T.branch_name from branch T where  
T.assets > some (select s.assets from branch s where s.branch-  
city='ong');
```

```
O/P::: branch_name  
erd  
gun  
crd  
suj
```

24. Find all the names of all the branches that have assets greater than that of each branch in ong?

SQL> select T.branch_name from branch T where T.assets>all(
select s.assets from branch s where s.branch-city='ong');

O/P:::branch_name
erd
gun
crd
subj

25. Find all the branches that have height average balance?

SQL>select branch_name from account groupby branch_name
having avg(balance)>=all(select avg(balance)from account groupby
branch_name);

O/P::: branch_name
subj

26. Find the names of the sailors who have reserved the red or green boats?

SQL>select s.sailor_name from sailors s,reserves r,boats b where
s.sid=r.sid and r.bid=b.bid and b.bcolor='red';
union
select s.sailor_name from sailors s,reserves r,boats b where
s.sid=r.sid and r.bid=b.bid and b.bcolor='green';

O/P::: no rows selected.

27. Find the names of sailors who have reserved red and green boat?

```
SQL> select s.sailor_name from sailors s,reserves r,boats b where
s.sid=r.sid and r.bid=b.bid and b.bcolor='red';
      intersect
      select s.sailor_name from sailors s,reserves r,boats b where
s.sid=r.sid and r.bid=b.bid and b.bcolor='green';
```

O/P::: no rows selected.

28. Find the names of sailors who have reserved red boat but not green boat?

```
SQL> select s.sailor_name from sailors s,reserves r,boats b where
s.sid=r.sid and r.bid=b.bid and b.bcolor='red';
      minus
      select s.sailor_name from sailors s,reserves r,boats b where
s.sid=r.sid and r.bid=b.bid and b.bcolor='green';
```

O/P::: no rows selected.

29. Find all sid's of sailors who have a rating of 5 or reserved the boat 6?

```
SQL>select s.sid from sailors s where s.raring=5
      union
      select r.sid from reserves r where r.nid=6;
```

O/P::: sid
3
6

30. Display the minimum rating value from sailors table.

```
SQL>select MIN(s.rating) from sailors s;
```

O/P:::MIN(s.rating) 4

31. Find the average ages of all sailors?

SQL>select avg(s.age)from sailors s;

O/P::: avg(s.age)
18.75

32. Find the average age of sailors with rating of 5?

SQL>select avg(s.age)from sailors s where s.rating=5;

O/P::: avg(s.age)
18

33. Count the no.of sailors?

SQL> select count(*) from sailors;

O/P::: count(*)
4

34. Count the no.of different sailors?

SQL> select count(distinct sname) from sailors;

O/P:::count(distinct sname)
4

35. Find the age of youngest sailors for each rating level?

SQL>select s.rating,min(s.age)from sailors s groupby s.rating;

O/P::: rating min(s.age)
4 20
5 18
6 19

36. Find the age of youngest sailors who is eligible to vote for each rating level with at least 2 such sailors?

```
SQL>select s.rating,min(s.age)from sailors s groupby s.rating  
having count(*)>1;
```

O/P::: no rows selected.

37.Find all the customers who have both an account and a loan at the bank?

```
SQL>select customer_name from borrower B where exists(select *  
from depositor d where d.customer_name=B.customer_name);
```

O/P:: customer_name

pav
roh
raji
paddu
neeru

UPDATION QUERIES

1. Supports that annual interest payments are being made and all balances are to be increased by 5%?

SQL>update account set balance=balance+(balance*0.05);

O/P:::5 rows updated

2. Suppose that accounts balances \$10000 receives 10% interest?

SQL>update account set balance=balance+(balance*0.1)where
balance>=10000;

O/P:::2 rows updated.

3. Pay 5% interest on accounts balance whose balance is greater than average?

SQL> update account set balance=balance+(balance*0.05)where
balance>(select avg(balance) from account);

O/P:::2 rows updated.

DELETION QUERIES

1. Delete all the tuples from the loan relation?

SQL>delete from loan;

O/P:::5 rows deleted.

2. Delete all of smiths account records?

SQL>delete from depositor where customer_name="smiths";

O/P:::1 rows deleted.

3. .Delete all loan amounts between 1300 and 1500?

SQL>delete from loan where amount between 1300 and 1500;

O/P:::3 rows deleted.

4. Delete all amounts at every branch located in prd?

SQL>delete from account where branch_name in(select branch_name from branch where branch_city='prd');

O/P:::no rows deleted.

5.Delete all amounts with balances below the average at the branch?

SQL>delete from account where balance(select avg(balance)from account);

O/P:::3 rows deleted.

VIEWS

1. **SQL>** Create a view v1 as select sname, age, rating from sailors where rating>5

Output: View Created

SQL>save v1.sql

Output: Created file v1.sql

SQL>select * from v1;

Output:	Sname	age	Rating
	Aaa	25	6
	Bbb	30	7
	Ccc	35	10

SQL>Update v1 set age=26 where rating=10 and sname='ddd'

Output: 1 row updated

SQL>select * from v1;

Output:	Sname	age	Rating
	Aaa	25	6
	Bbb	30	7
	Ccc	35	10

2.SQL> Create view v2 (rate, minimum) as select rating, min(age) from sailors group by rating;

Output: View Created

SQL> Select * from v2;

Output: Rate Minimum

5	18
6	22
7	52
10	21

SQL>Update V2 set minimum=19 where rate=5;

Output: date manipulation operation not legal on this view

3.SQL> Create view v3 (sname, bid) as select sname, bid from sailors s reserves r where s.sid=r.sid and r.day='09-may-07';

Output: View Created

SQL>sname v3.SQL

Output: create file v3.SQL

SQL> select * from v3

Output: Sname Bid
 dd 4

4. SQL> Create view v4 as select s.sname, s.rating from sailors s, reserves r boats b where s.sid=r.sid and r.bid=b.bid and b.bcolor='red';Union Select s.sname, s.rating from sailors s, reserves r, boats b where s.sid=r.sid and r.bid=b.bid and b.bcolor='green';

Output: View Created

SQL> select * from v4

Output: Sname Rating

Aaa	5
Ccc	7
Ddd	10

SQL>Update v4 set rating=25, where sname='aaa';

Output: data manipulation operation not legal on this view

SQL>save v4.SQL

Output: created file v4.SQL

5.SQL> Create view v5 as select s.rating t1 as san r, s.sid from sailors s, reserves r where r.bid>r2.bid and r1.day=r2.day and r1.sid=s.sid and s.sid=r2.sid.

Output: View created

SQL> select * from v5;

Output: Rank sid

3	101
4	102
5	103
8	106

SQL> save v5.SQL

Output: created file v5.SQL

SQL> uptade v5 set rank=2 where sid=101;

Output: virtual coloumn not allowed here

PL/SQL

1. Display whether the given number is Even or Odd.

```
Declare
    N number(5):=&N;
Begin
    dbms_output.put_line(N);
    If mod(N,2)=0 then
        dbms_output.put_line(N || "is even");
    Else
        dbms_output.put_line(N || "is odd");
    End if;
End;
/
```

input :-

N=5

output:-

5 is odd

input :-

N=6

output:-

6 is even

2. Calculate Factorial for the given number.

Declare

```
n      number(5):=&n;  
fact   number(5):=1;
```

Begin

```
dbms_output.put_line(n);  
while(n<=0)  
loop  
    fact:=fact*n;  
    n:=n-1;  
end loop;  
dbms_output.put_line("factorial of " n ||" is " fact);
```

end;

/

input :-

n=3

output:-

factorial of 3 is 6

input :-

n=5

output:-

factorial of 5 is 120

3. Find the largest number among three numbers.

Declare

n1 number(3):=&n1;

n2 number(3):=&n2;

n3 number(3):=&n3;

Begin

dbms_output.put_line(n1);

dbms_output.put_line(n2);

dbms_output.put_line(n3);

if n1>n2 then

 if n1>n3 then

 dbms_output.put_line(n1 || "is largest
number");

 else

 dbms_output.put_line(n3 || "is largest
number");

 end if;

else

 if n2>n3 then

 dbms_output.put_line(n2 || "is largest
number");

 else

 dbms_output.put_line(n3 || "is largest
number");

 end if;

end if;

end;

/

input :-

n1=15

n2=20

n3=25

output:-

25 is the largest number

4. Check whether the given number is prime or not

Declare

n number(3):=&n;

I number(3);

count1 number(3):=0;

Begin

dbms_output.put_line(n);

for i in reverse

loop

if mod(n,i)=0 then

count1=count1+1;

end if;

end loop;

if count1=2 then

dbms_output.put_line(n || "is prime
number");

else

dbms_output.put_line(n || "is not a prime
number");

end if;

end;

/

input: -

n=5

output: -

5 is prime number

input: -

n=6

output: -

6 is not a prime number.

5. Find the Fibonacci series.

Declare

```
n    number(3):=&n;  
x    number(3):=0;  
y    number(3):=1;  
z    number(3);
```

Begin

```
dbms_output.put_line(n);  
dbms_output.put_line(x);  
dbms_output.put_line(y);  
z:=x+y;  
dbms_output.put_line(z);  
n:=n-z;  
while(n>0)  
loop  
    x:=y;  
    y:=z;  
    z:=x+y;  
    n:=n-1;  
    dbms_output.put_line(z);  
end loop;  
end;  
/
```

input :-

n=5;

output:-

0 1 1 2 3 5

6. Print the prime numbers in the given range.

Declare

```
p    number(3):=&p;  
c    number (3):=0;  
i    number(3):=1;
```

Begin

```
While(i<=p)
```

```
Loop
```

```
    If mod(p,i)=0 then
```

```
        c=c+1;
```

```
    end if;
```

```
    If c=2 then
```

```
        dbms_output.put_line(i ||);
```

```
    end if;
```

```
end loop;
```

```
end;
```

```
/
```

input: -

p=20

output: -

2 3 5 7 11 13 17 19

7. Write a program to reverse a number

Declare

```
n    number(3):=&n;  
n1   number(3);  
m    number(3):=0;
```

Begin

```
dbms_output.put_line(n);  
n1:=n;  
while n1>0  
loop  
    m:=m*10+mod(n,10);  
    n1:=n1/10;  
end loop;  
dbms_output.put_line(m);
```

end;

/

input :-

123

output:-

321

8. Program to implement string reversal.

```
Declare
  S    varchar2(10):=&s;
  Revs varchar2(10);
  Strlen  number(2);
Begin
  dbms_output.put_line(s);
  strlen:=length(s);
  for I in 1..strlen
  loop
    revs:=revs || substr(s,strlen,1);
    strlen:=strlen-1;
  end loop;
  dbms_output.put_line(revs);
end;
/
```

input :-

s= abc

output:-

cba

9. Program to find the number of even and odd numbers and the sum of each.

Declare

```
R    number(3):=&n;  
Se   number(10):=0;  
so   number(10):=0;  
cte  number(5):=0;  
ctu  number(3):=0;
```

Begin

```
dbms_output.put_line(n);  
for i in 1..n loop  
    if mod(i,2)=0 then  
        cte:=cte+1;  
        se:=se+i;  
    else  
        cto:=cto+1;  
        so:=so+i;  
    end if;  
end loop;  
dbms_output.put_line("sum of even numbers " se ||);  
dbms_output.put_line("No. of even numbers " cte ||);  
dbms_output.put_line("sum of odd numbers " so ||);  
dbms_output.put_line("No. of odd numbers " cto ||);
```

end;

/

Input :-

n=4

Output:-

```
cte : 2  
cto : 2  
se  : 6  
so  : 4
```

10. Program to check whether the given no. is Armstrong or not.

```
Declare
    N    number(3):=&n;
    Tempnumber(3);
    Sum1 number(3);
    I    number(3):=0;
Begin
    While(n>0)
    Loop
        I:=mod(n,10);
        Sum1:=sum1+(I*I*I);
        N:=floor(n/10);
        If temp=sum1 then
            dbms_output.put_line(temp || "is Armstrong
number " );
        else
            dbms_output.put_line(temp || "is not a
Armstrong number " );
        end if;
    end;
/
```

Input :-

n=153

Output: -

153 is Armstrong number

11. Write a program to invest area and radius of a circle.

Declare

```
Pi          number(4,2):=3.14;  
Radius      number(5);  
Area        number(14,2);
```

Begin

```
Radius:=3;  
While radius<=7  
Loop  
    Area:=pi*power(radius,2);  
    Invest into area values(radius,area);  
    Radius:=radius+1;  
End loop;
```

End;

/

Output: -

Radius	Area
3	28.26
4	5.24
5	78.50
6	113.04
7	153.86

12. Write a program to find whether a number is divisible by 9 or not.

Declare

```
N    number(3):=&n;
```

Begin

```
    If mod(n,9)=0 then
        dbms_output.put_line(n||"is divisible by 9");
    else
        dbms_output.put_line(n||"is not divisible by 9");
    end if;
end;
/
```

Input: -

n=36

Output: -

36 is divisible by 9

Input: -

n=35

Output: -

35 is not divisible by 9

CURSOR

1. Declare
Cursor c1 is
Select loan amt, loan no into amt no from loan;
Begin
Open c1
Fetch c1 loan amt, loan no into amt, no
Loop
If amt>20000
Then
Insert into best loan values(loan amt 1, act no, bname);
Else
Insert into coorst loan values(loan amount 1, acc no, bname);
End if;
End;
2. Declare
Cursor c1 is
Select cid, salary from employee where salary>30000;
E1 employee.eid%type;
E2 employee.salary%type;
E3 employee.salary%type;
Begin
Open c2
Fetch c2 cid, salary into c1,c2
Loop
Update employee set salary=salary+(10/100*c2);
C3=c2+(c2*10/100);
Insert into increment values (e1,e3);
End loop
Close c2
End

TRIGGERS

AIM: ABOUT TRIGGERS

List of Values (LOVs)

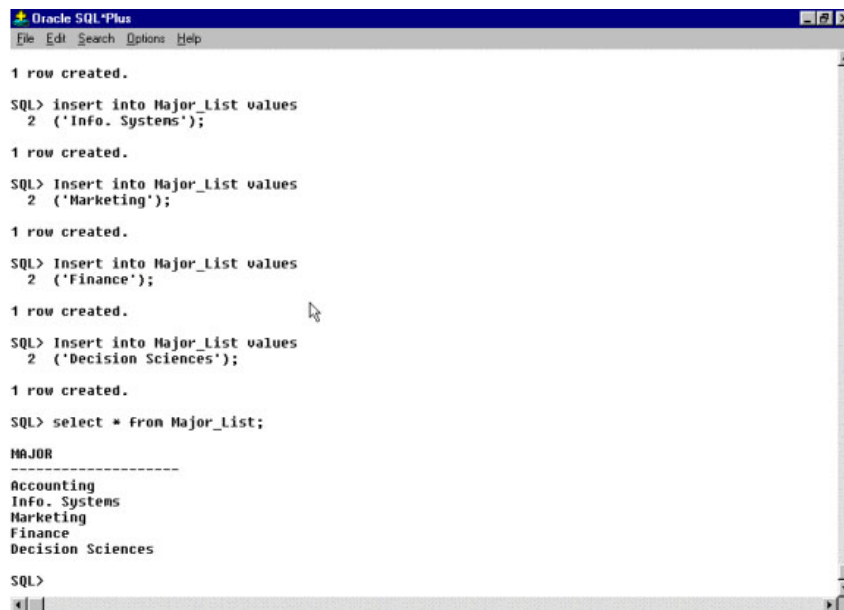
1. In this form we are designing, we will create a List of Values for major, so that the user can select a major from the list. To do this, first we will have to create table called Major_List.
2. To do this, go to SQL*Plus and type the following code at the SQL prompt

```
SQL> CREATE TABLE MAJOR_LIST  
2 (MAJOR VARCHAR2(20));
```

3. Now type in the following statement to insert more data into the Major_list table:

```
SQL> INSERT INTO MAJOR_LIST VALUES ('Accounting');  
SQL> INSERT INTO MAJOR_LIST VALUES ('Info. Systems');  
SQL> INSERT INTO MAJOR_LIST VALUES ('Marketing');  
SQL> INSERT INTO MAJOR_LIST VALUES ('Finance');  
SQL> INSERT INTO MAJOR_LIST VALUES ('Decision Sciences');
```

4. Now, type in Select * from Major_List to view your newly created table and its records. (See Figure 9.1)



```
Oracle SQL*Plus  
File Edit Search Options Help  
  
1 row created.  
  
SQL> insert into Major_List values  
2 ('Info. Systems');  
  
1 row created.  
  
SQL> Insert into Major_List values  
2 ('Marketing');  
  
1 row created.  
  
SQL> Insert into Major_List values  
2 ('Finance');  
  
1 row created.  
  
SQL> Insert into Major_List values  
2 ('Decision Sciences');  
  
1 row created.  
  
SQL> select * From Major_List;  
  
MAJOR  
-----  
Accounting  
Info. Systems  
Marketing  
Finance  
Decision Sciences  
  
SQL>
```

Figure 9.1: Creating table Major_List in SQL*PLUS

5. First, select LOV in the Object Navigator on Developer/2000 and click on the "+" sign on the toolbar to your left. (See Figure 9.2)

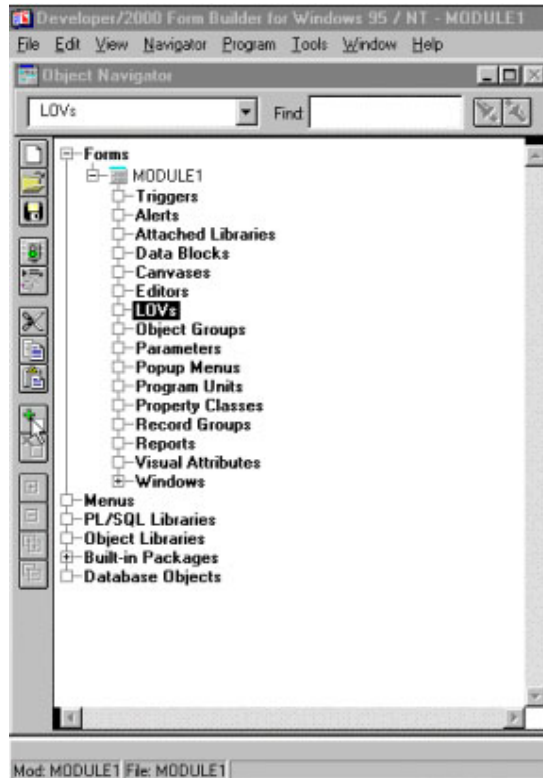


Figure 9.2: Creating a new LOV

6. Immediately, the window for the new LOV will pop up. (See Figure 9.3)

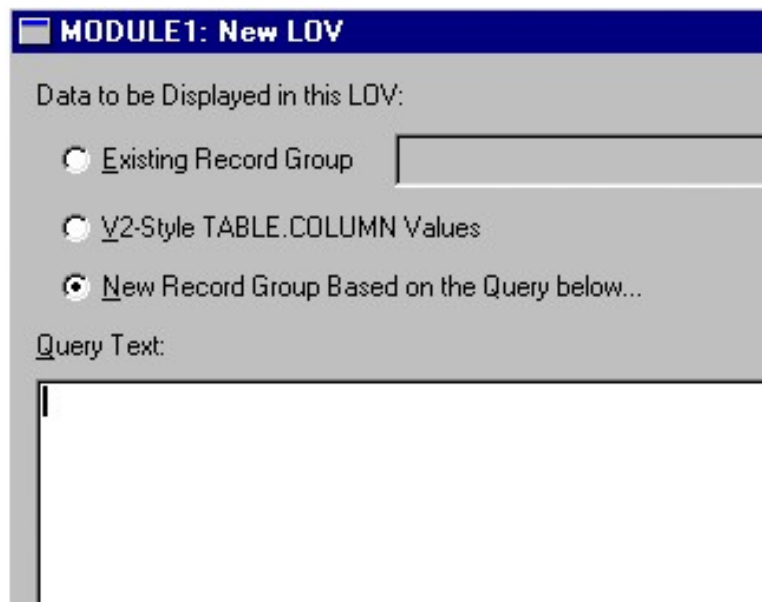


Figure 9.3: The New LOV Window

7. Type in the following Select statement in the window to connect the new LOV to the Major_List table that we created earlier and click OK.
(See figure 9.4)

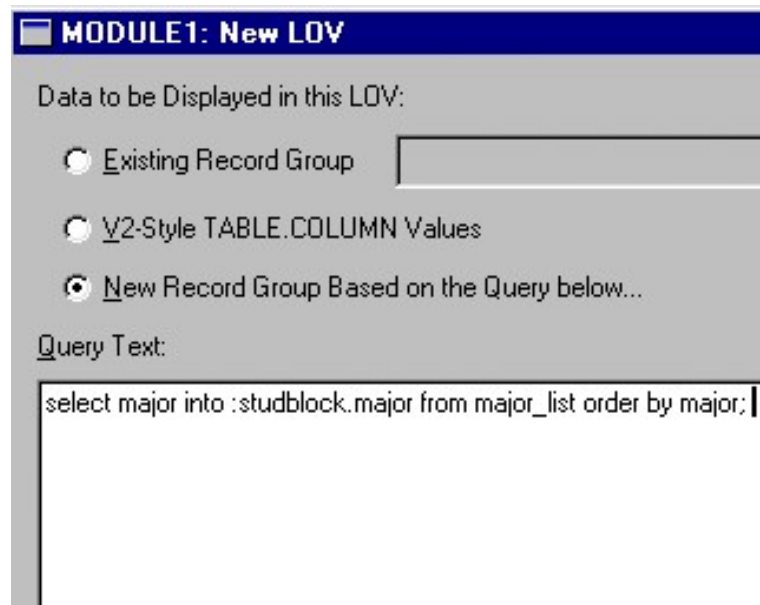


Figure 9.4: The New LOV Window with the select code

select major into :studblock.major from major_list order by major;

8. Once you click OK, you will come back to the Object navigator window. Now, name the LOV as MAJOR_LOV by double clicking the word LOV.
(See Figure 9.5)

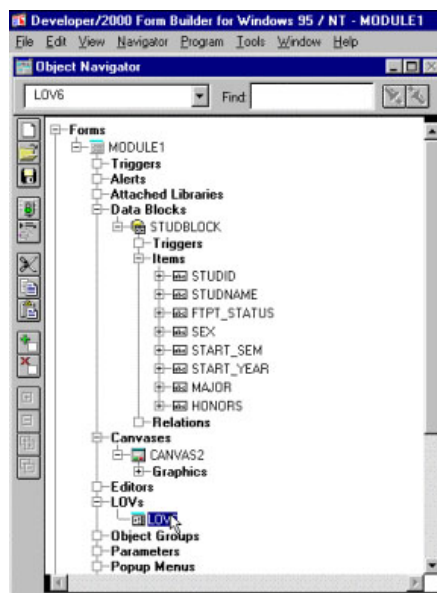


Figure 9.5: Changing the name of the LOV

9. Now go back to the Canvas View by clicking on the small picture icon just beneath the word Canvases in the Object Navigator. (See Figure 9.6)

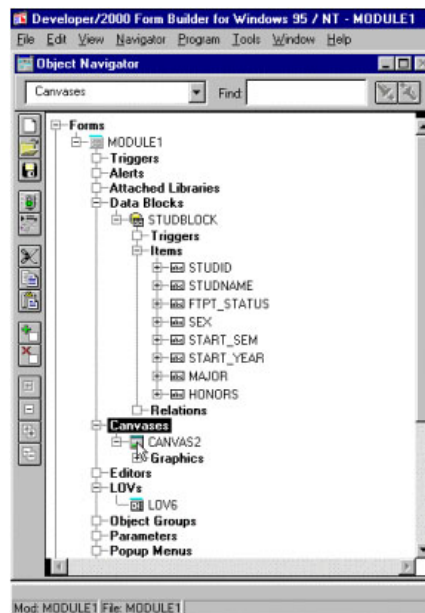


Figure 9.6: Returning to the Canvas view

10. In the Canvas View, we will create a push button and position it right beside the Major data field. To create the push button, select the box like icon from the toolbox, click on it once and drop it beside the Major data field. (See Figure 9.7)

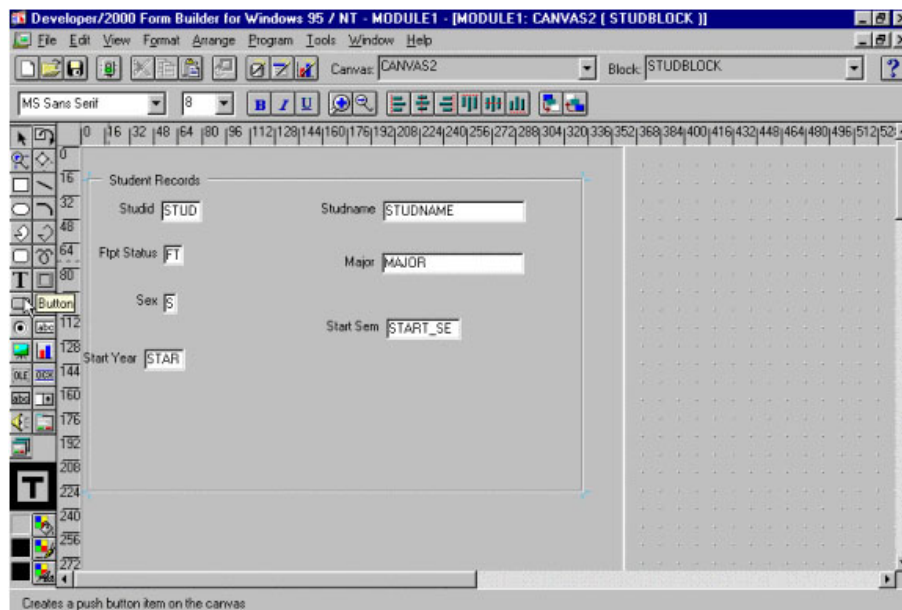


Figure 9.7: Creating a push button to place it beside the Major data field

11. After the push button has been created, then right click on the button and select Property Palette from the list that pops up. (See Figure 9.8)

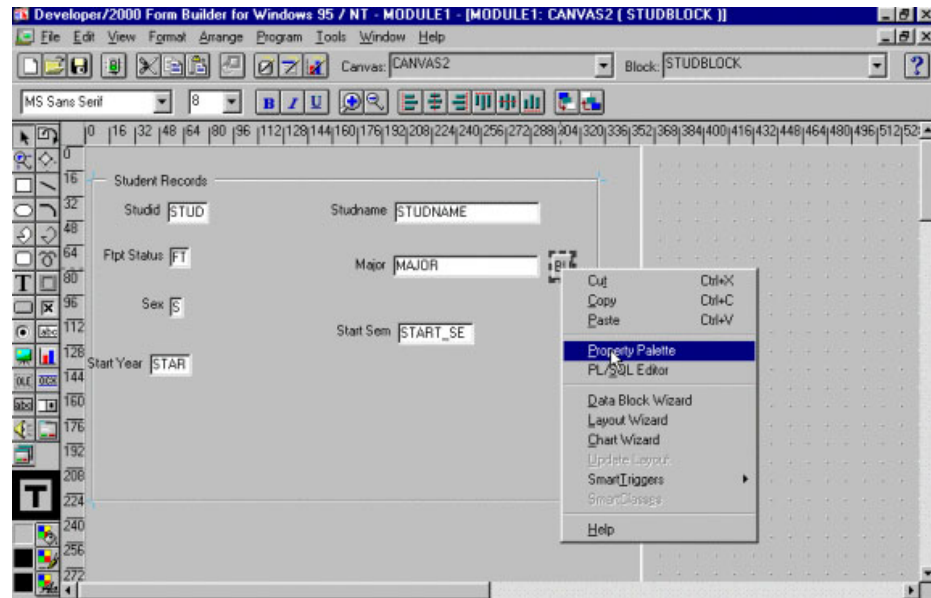


Figure 9.8: Selecting Property Palette for the Push Button

12. Once you are in the Property Palette, remove the label, specify Iconic as Yes and in the icon Filename type Down. Click on the close button in the lower taskbar located on the upper right corner. (See Figure 9.9)

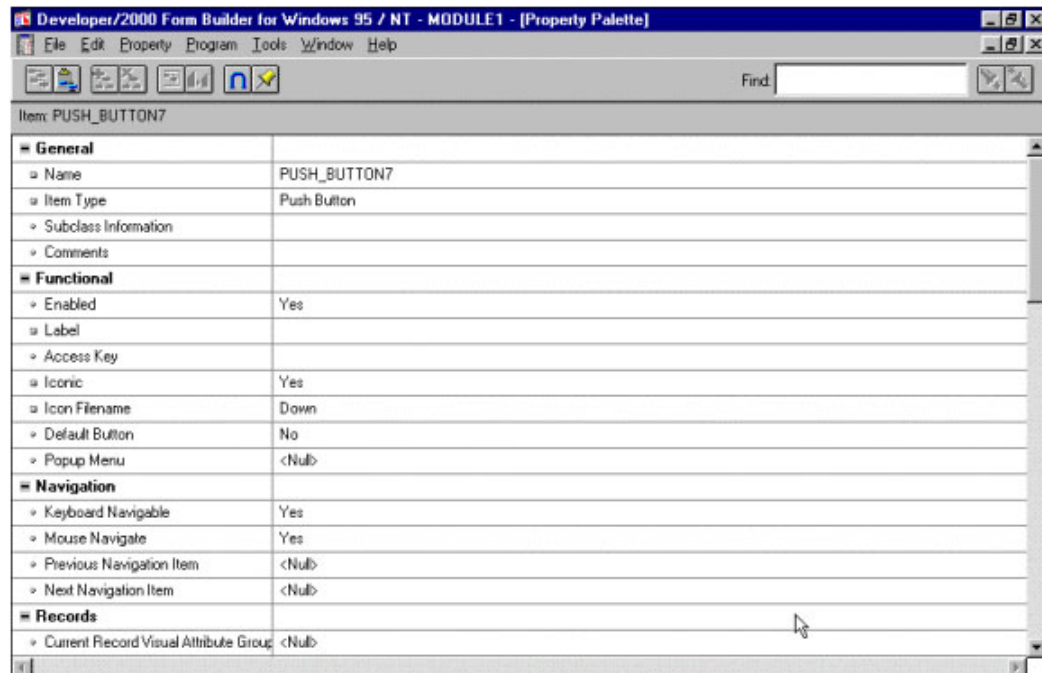


Figure 9.9: The Property Palette for the Push Button

13. You will now return to Canvas View. Your Canvas will now look like Figure 9.10.

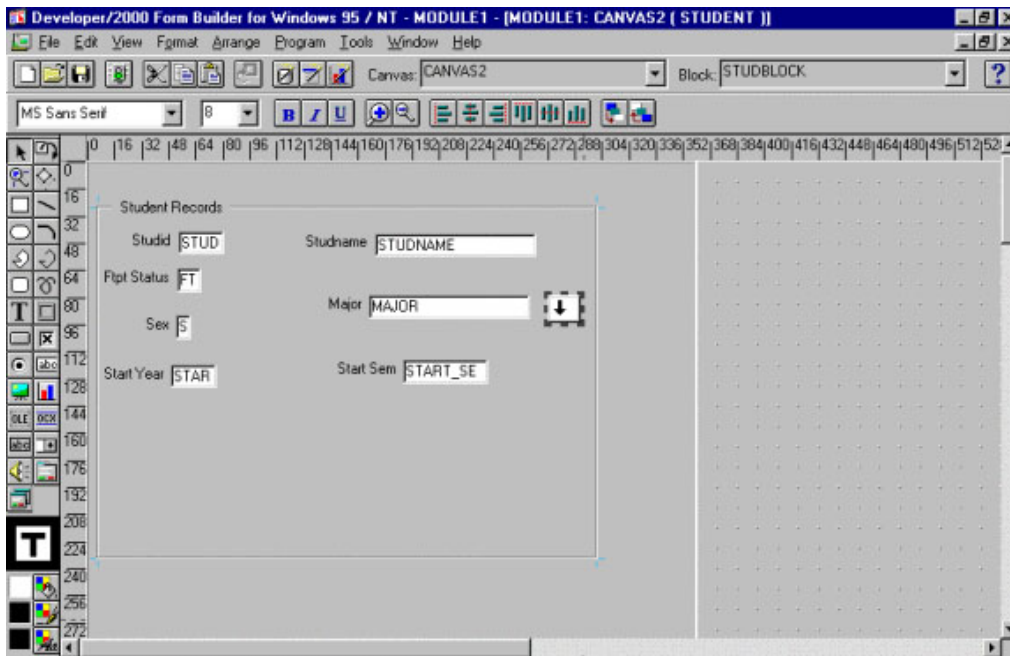


Figure 9.10: The Canvas View with the Push Button for the LOV

14. Now back in the canvas, right click on push and this time select the PL/SQL Editor. (See Figure 9.11)

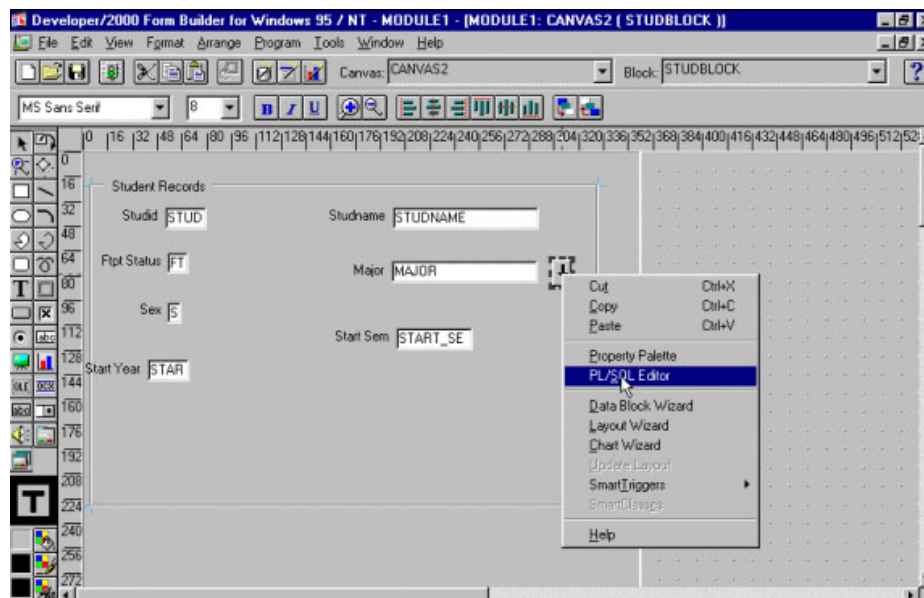


Figure 9.11: Selecting the PL/SQL Editor of the Iconic Button

15. In the PL/SQL Editor we will write a trigger that will connect this button to the table called *Major_List*, so that when the user clicks on the button they will be able to view the list of options. When you select the PL/SQL Editor, the window for the new trigger selection will appear. (See Figure 9.12)

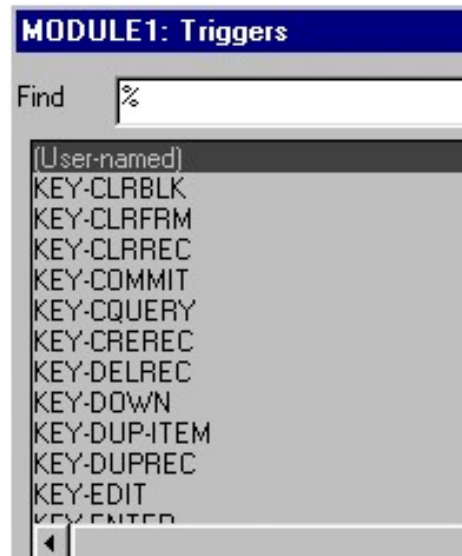


Figure 9.12: The Window with the list of triggers available to the user

16. We will write a When-Button-Pressed trigger, since we would like the code to be activated when the user presses the button. Scroll down and select When-Button-pressed trigger. (See Figure 9.13)

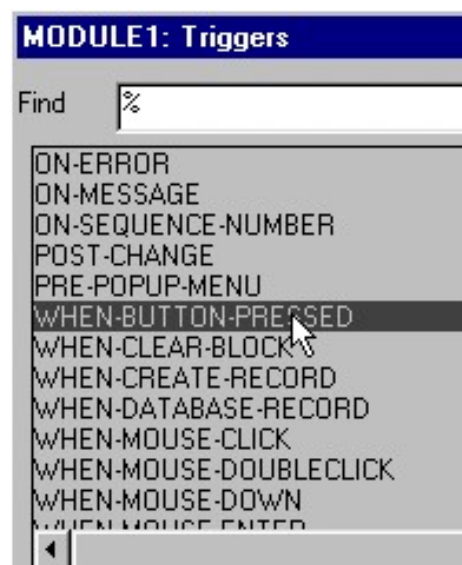


Figure 9.13: Selecting the When-Button-Pressed trigger

17. Immediately, you will see the PL/SQL Editor window. (See Figure 9.14).
Type in the following code in the blank space in the Editor.

```
Declare
Return_LOV Boolean;
Begin
Return_LOV :=show_LOV('Major_LOV');
End;
```

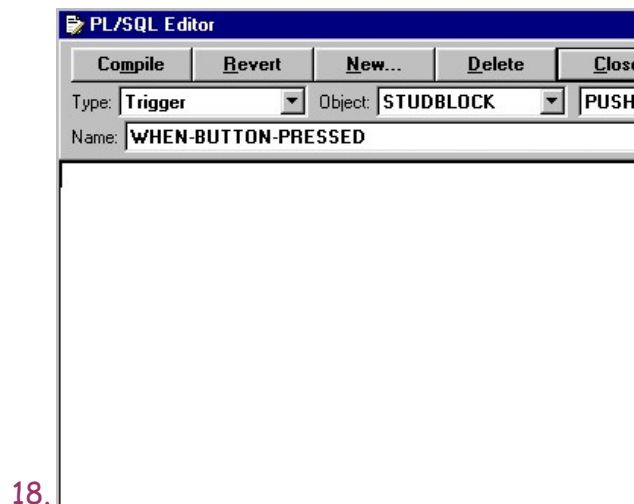
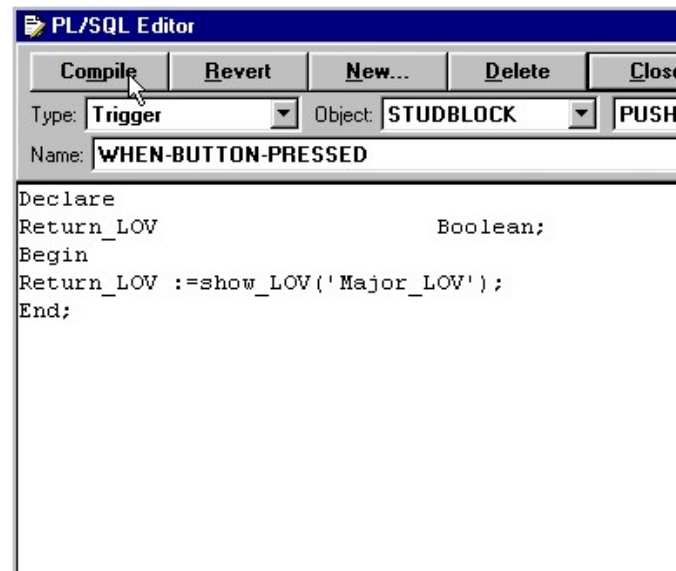


Figure 9.14: The PL/SQL Editor window

18. After typing in the code, click compile on the upper left-hand corner of the window. (See Figure 9.15)



19. To test how the button works, you can view the form by returning to the Canvas View and selecting Run Form from the Program menu. (See Figure 9.16)

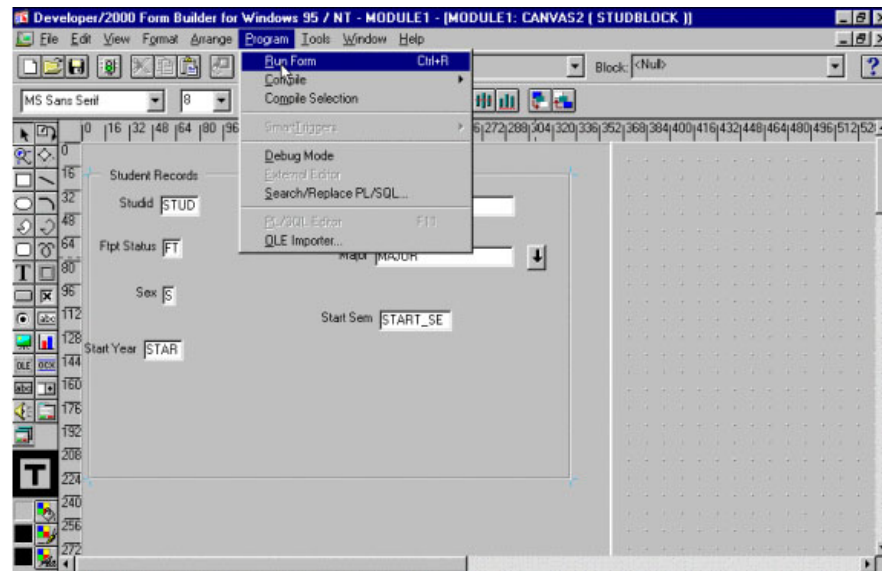


Figure 9.16: Selecting Run Form from the Program menu

20. When the form is running, type in an ID number in the STUDID field (there has to be data in the primary field). Then click the button with the down arrow. The list of majors will pop up. (See Figure 9.17). To return to the canvas View, simply click OK in the list of majors and click the close button in the upper right taskbar.

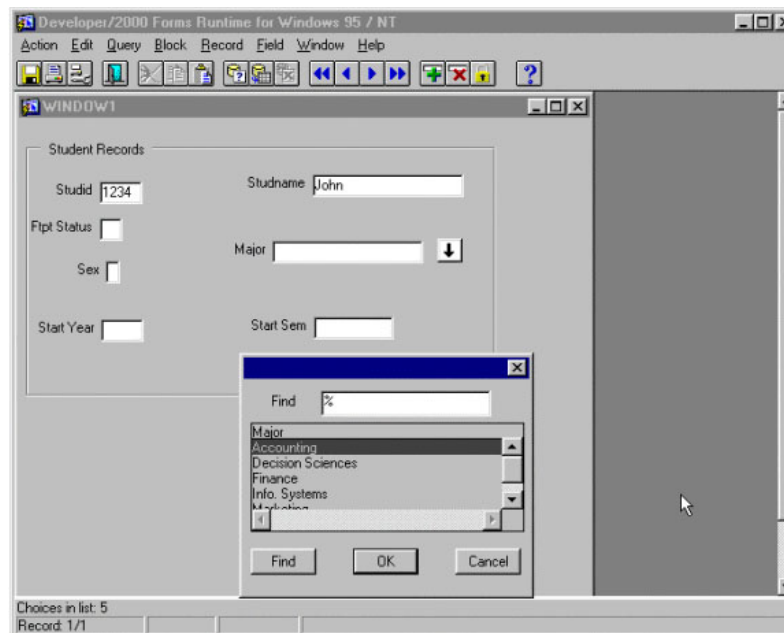


Figure 9.17: The List of values for the Majors

21. We will now create radio buttons for the FTPT_Status to give the user an option of creating either Full-Time or Part-Time.
22. To do this, right click on the FTPT_Status field in the canvas view and go to its Property palette. (See Figure 9.18)

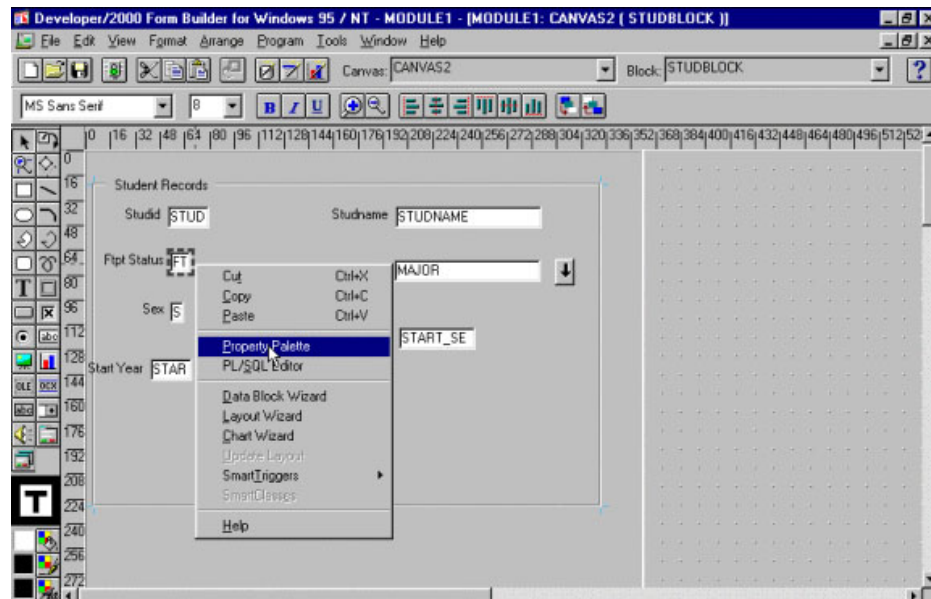


Figure 9.18: Selecting the Property Palette for the FTPT_Status field

23. Change the item type from text to Radio Group and set the initial value to either FT or PT. Close the Property Palette by clicking the close button on the upper right hand corner. (See figure 9.19)

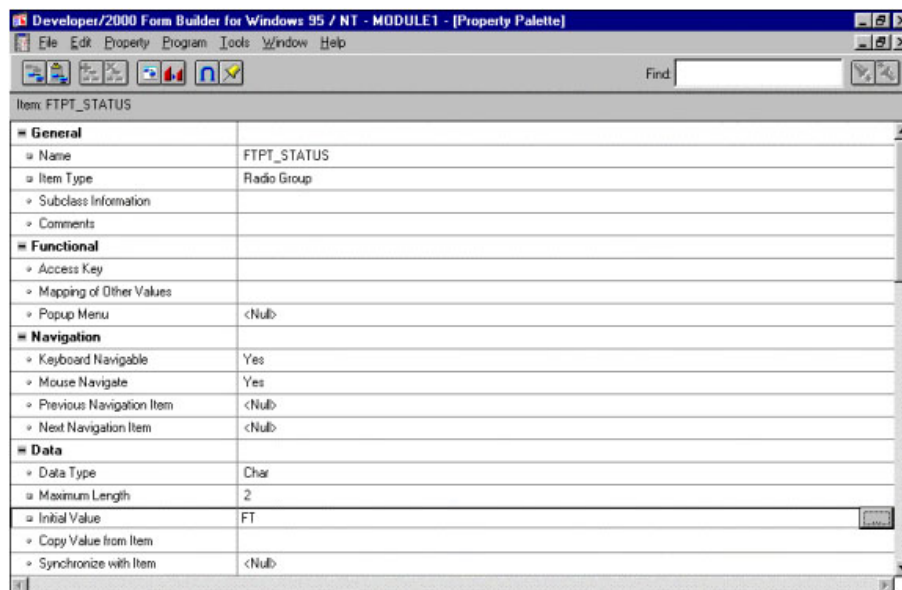


Figure 9.19: The Property Palette for FTPT_Status

24. Once you return back to the canvas, you will realize that the data field for the FTPT_Status is no longer visible. This is because we have specified it to be a radio group and not a text item. To see them, we will have to insert radio buttons. Select radio button from the tool palette on your left. (See Figure 9.20)

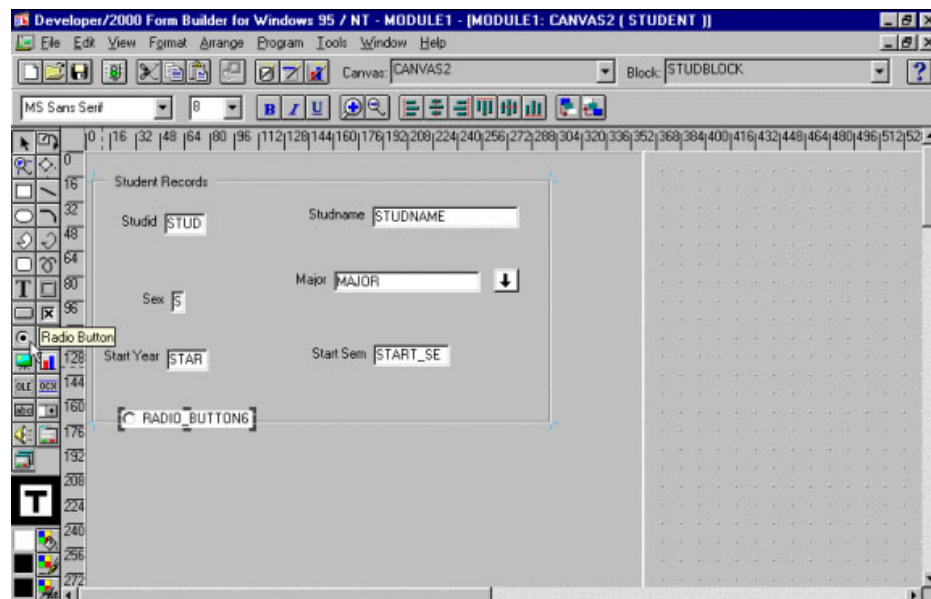


Figure 9.20: The palette with the icon for the radio buttons

25. Drop the radio button into the canvas. Immediately a window will appear, prompting you to select the radio group you would like to attach this radio button to. Select the radio group FTPT_Status. (See Figure 9.21)

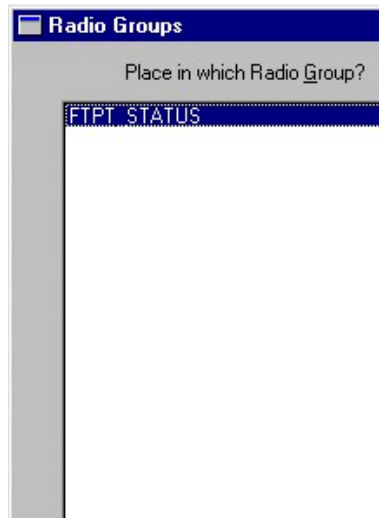


Figure 9.21: The Window for Attaching the radio button to a radio group

26. Now right click on the radio button and go to its Property Palette. Change the label of the button to 'Full_Time', change background color to gray and give the radio button a value of FT. (See Figure 9.22)

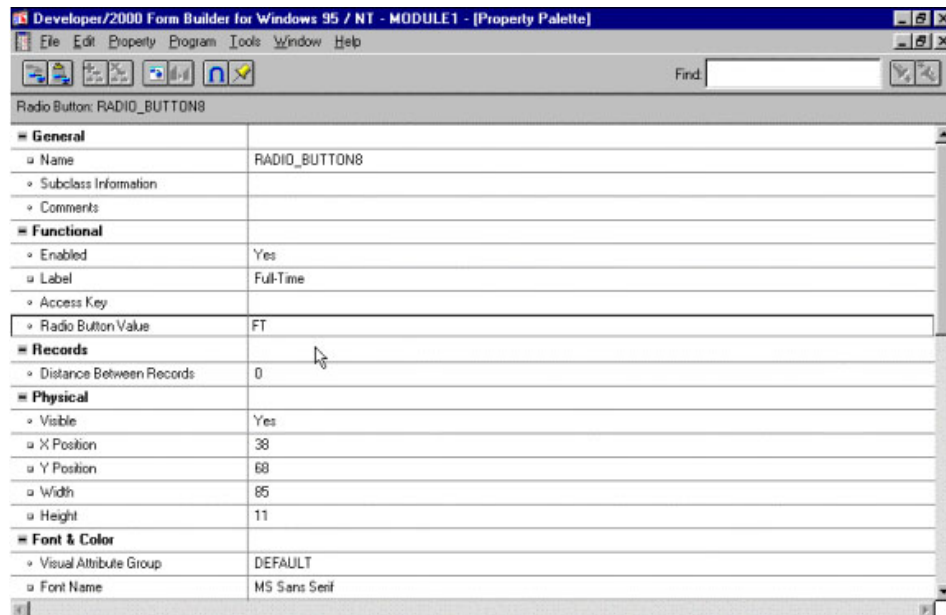


Figure 9.22: The Property Palette for the radio button

27. In the same manner create a second radio button, attach it to the radio group FTPT_Status, give it a label of 'Part-Time' and a value of PT.
28. Select a rectangle from the palette and draw it around the radio buttons. (See Figure 9.23)

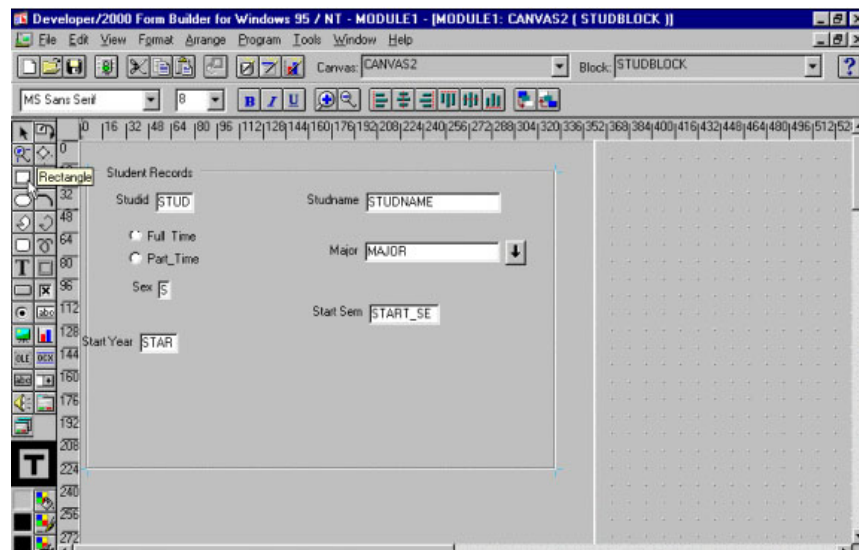


Figure 9.23: The Rectangle Icon in the palette

29.Go to the properties for the rectangle frame by right clicking on it, and change the fill pattern to none. (See Figure 9.24)

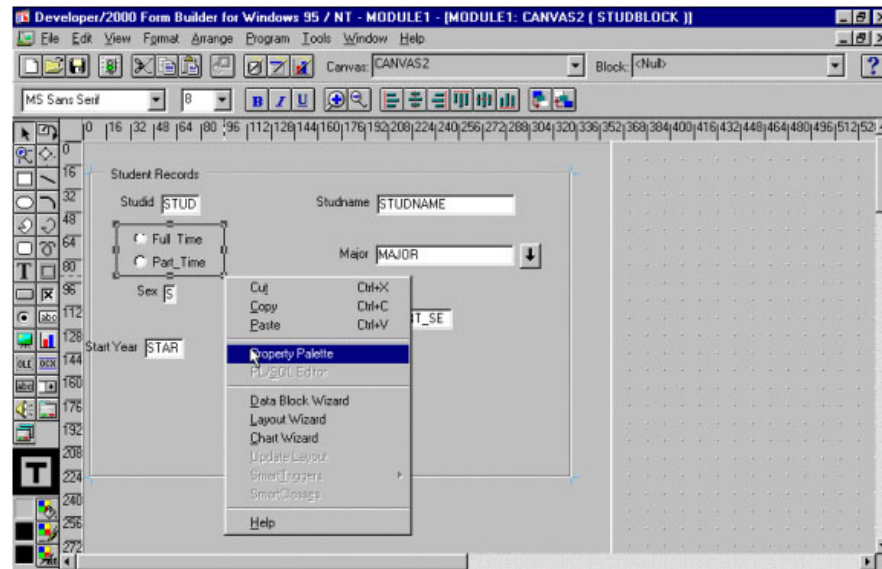


Figure 9.24: Selecting the Property Palette for the rectangle frame

30.Now, click on the rectangle frame and select Format→Bevel→Lowered to format the frame. (See Figure 9.25)

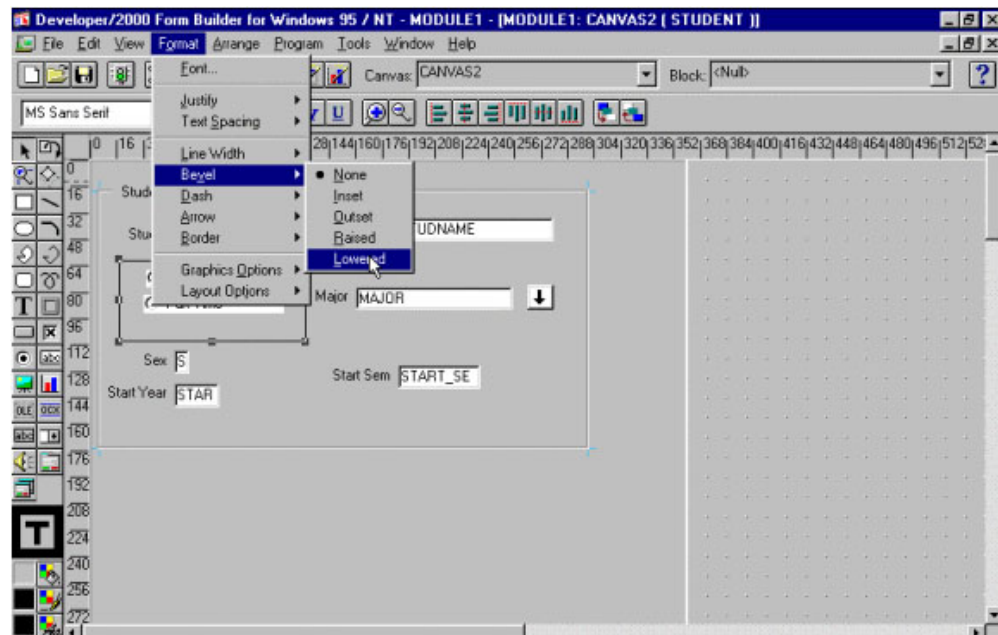


Figure 9.25: Formatting the Rectangle Frame around the radio buttons

Developer/2000 Forms Runtime for Windows 95 / NT

Action Edit Query Block Record Field Window Help

WINDOW1

Student Records

Studid Studname

☒ Full-Time
☐ Part-Time

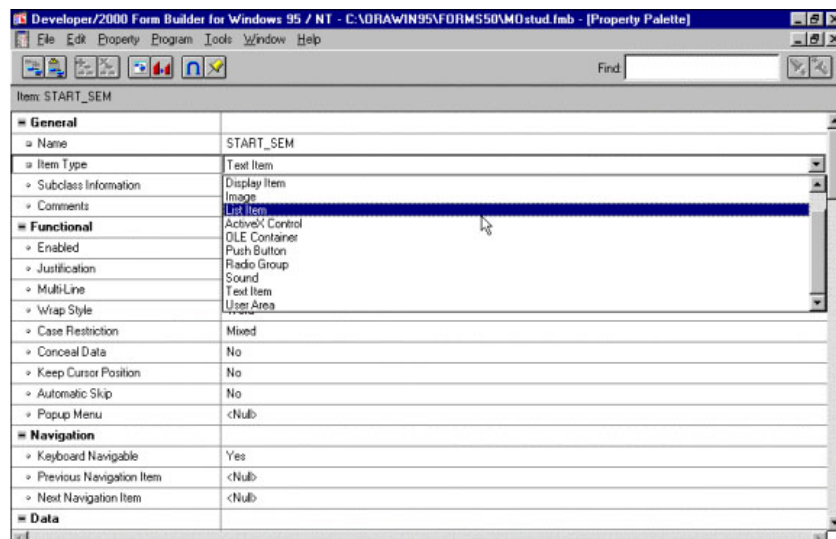
Major

Sex

Start Year Start Sem

Record: 1/1

32. We will now create a drop-down poplist for the variable `Start_Sem` that will contain four entries: Fall, Spring, Summer 1 and Summer 2. To do this, go to the Property Palette for the `Start_Sem` and change its item type from text to list items, and select type of list as poplist. (See Figure 9.27). Close the Property Palette to return to the Canvas View by clicking on the close button in the upper right hand corner



DATA BASE MANAGEMENT SYSTEM LAB

33. Click on the List Elements tab and type in the list elements and list values. In this lesson, the list elements and values will be same and will be Fall, Spring, Summer 1 and Summer 2. (However, in the values, Summer1 and Summer2 should be typed in without spaces). (See Figure 9.28)

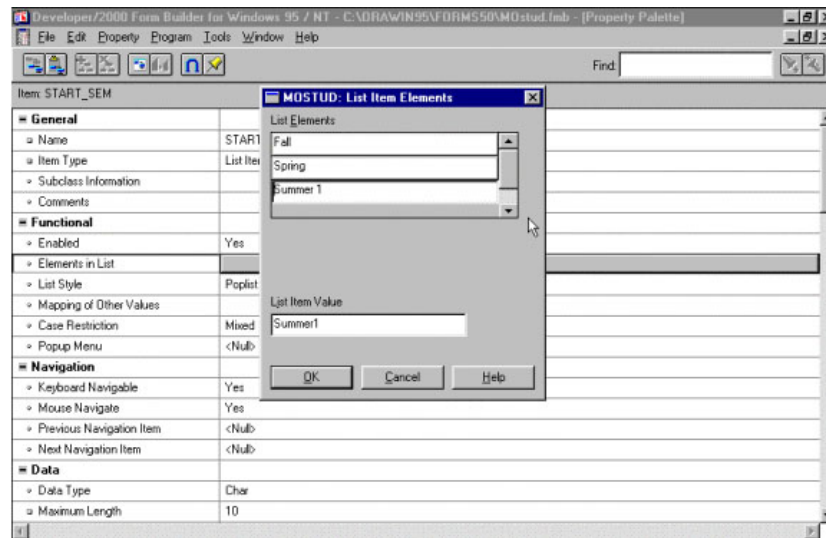


Figure 9.28: The Window for the List Elements

34. Now go to Programs → Run Form to view your newly created Poplist.

(See Figure 9.29)

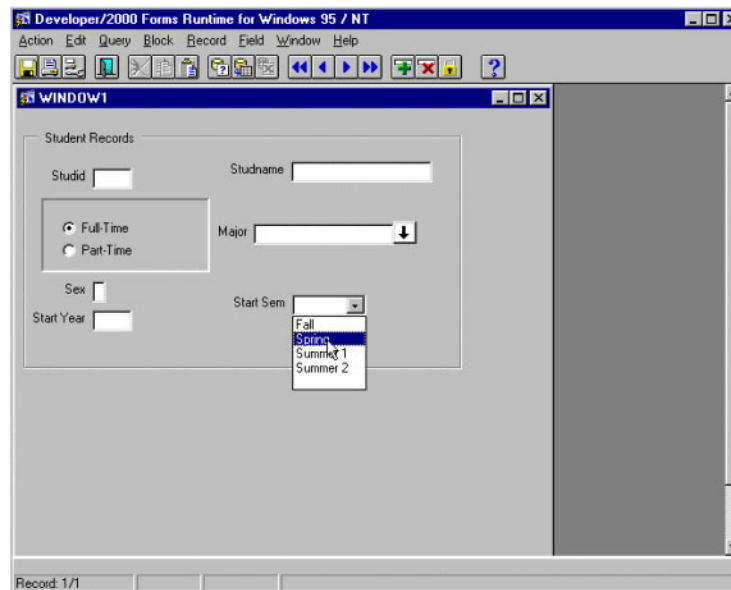


Figure 9.29: Window with the Poplist on Start_Sem

35. We will now create three push buttons and write triggers for each button. To create a push button, select the button icon from the palette on the left of the canvas and drop in the lower part of the form. (See Figure 9.30)

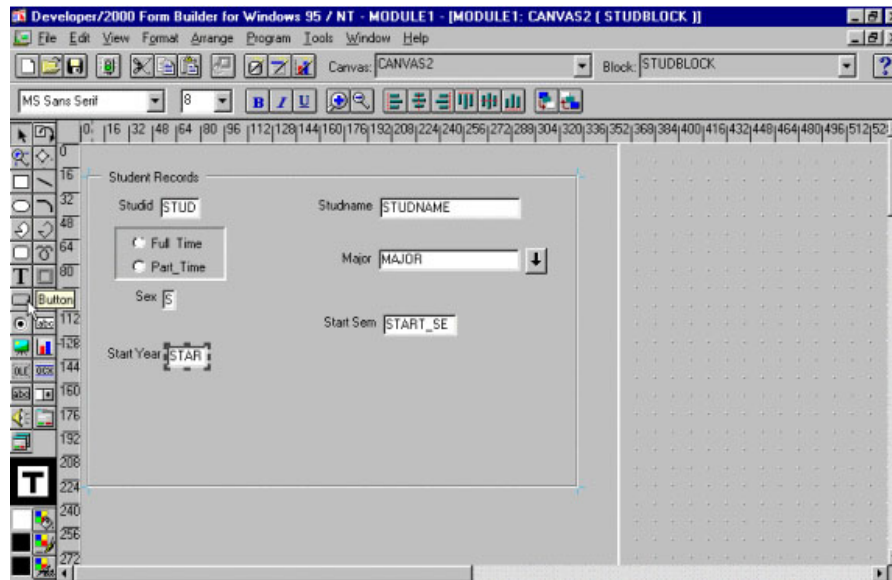


Figure 9.30: Selecting a push button

36. We will change the label of the first push button in the property palette to "Retrieve." To do this, right click on the button and go to its Property Palette. (See Figure 9.31).

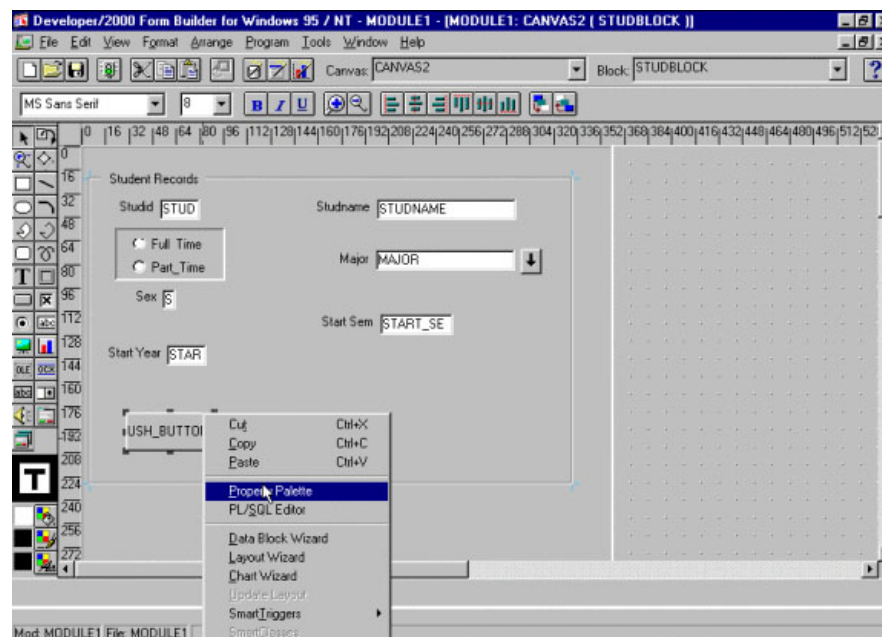


Figure 9.31: Selecting the Property Palette for the push button

37. Now select the PL/SQL editor by right clicking on the push button. (See Figure 9.32).

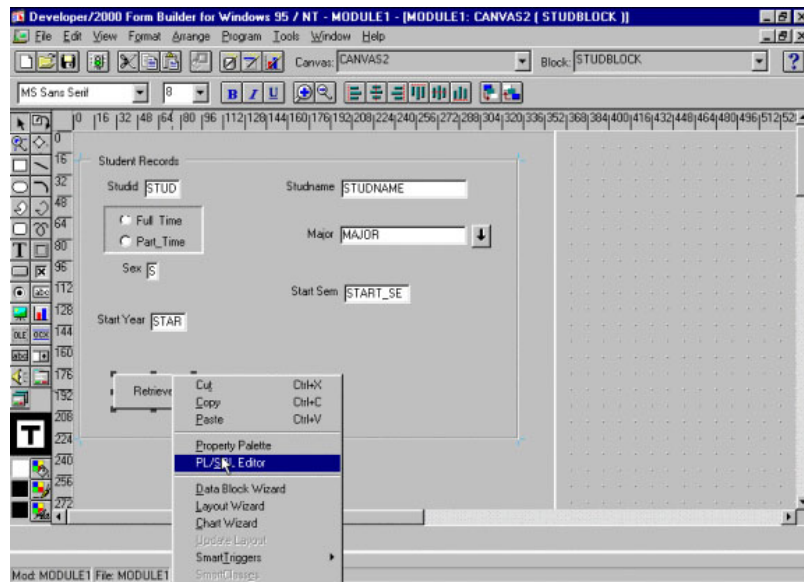


Figure 9.32: Selecting the PL/SQL Editor for the push button

38. Select the WHEN-BUTTON-PRESSED-TRIGGER, insert the following PL/SQL code in the blank space of the editor and then click Compile.

```
begin
select studid, studname, ftpt_status, sex, start_sem, start_year, major
into :studblock.studid, :studblock.studname, :studblock.ftpt_status,
:studblock.sex, :studblock.start_sem, :studblock.start_year,
:studblock.major
from student
where studid = :studblock.studid;
exception
when no_data_found then
message ('Invalid Student Id:Please enter a valid Id. ');
raise form_trigger_failure;
end;
```

39. In the above code, we are writing a select statement for retrieving the record of a student with any particular student ID. If no data is found on a particular student ID, then Developer/2000 will give an error message and raise the form_trigger_failure trigger. Run the form and type in a invalid STUDID to check the message. (See Figure 9.33)

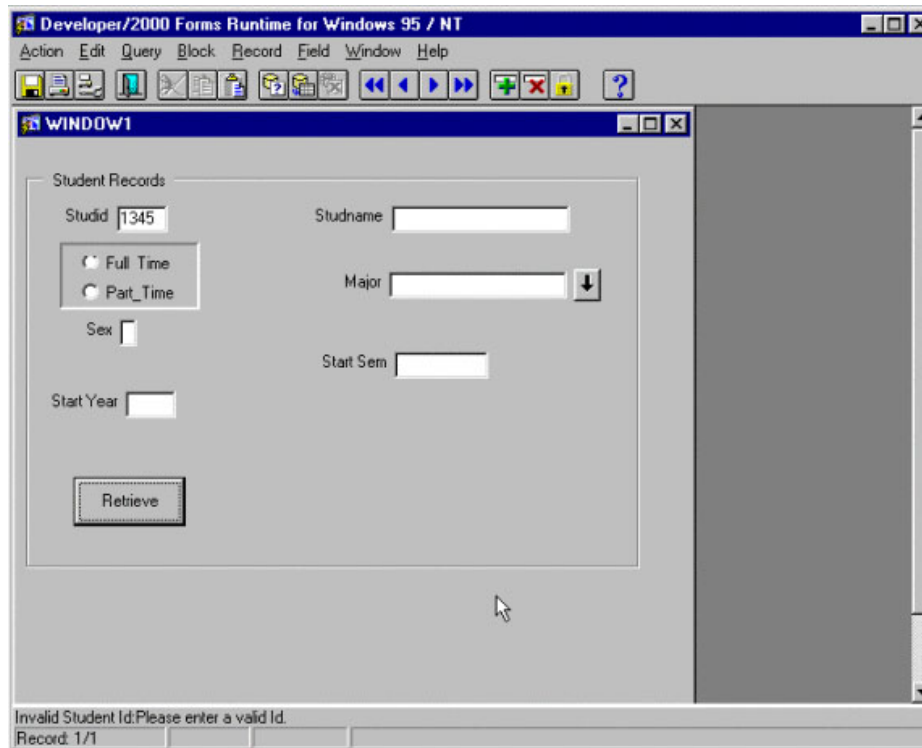


Figure 9.33: Error Message when invalid Student ID is entered by the user

In a similar manner, create two other buttons, totaling three push buttons.

40. Go to the second push button, right click on it and go to its property palette. Change its label to Insert. Now come back to the Layout Editor, right click again on the push button and go to its PL/SQL Editor. In it, write the following code in the When-Button-Pressed trigger:

```
Commit;  
Clear_Form;
```

41. Label the third push button as Clear. At the When-Button-Pressed Trigger, write the following code:

```
Clear_Form;
```

42. Now go back to the Canvas View and Program → Run Form to view your form with the three push buttons. (See Figure 9.34)

The screenshot shows a Windows 95-style application window titled "Developer/2000 Forms Runtime for Windows 95 / NT". The window has a menu bar with "Action", "Edit", "Query", "Block", "Record", "Field", "Window", and "Help". Below the menu bar is a toolbar with various icons. The main area of the window contains a form titled "Student Records". The form has the following fields and controls:

- Studid**: A text input field.
- Studname**: A text input field.
- Major**: A dropdown menu.
- Start Sem**: A dropdown menu.
- Sex**: A checkbox.
- Start Year**: A text input field.
- Full-Time**: A radio button.
- Part-Time**: A radio button.
- Retrieve**: A push button.
- Insert**: A push button.
- Clear**: A push button.

At the bottom of the window, there is a status bar that says "Record: 1/1".

Figure 9.34: The Form with the three Push buttons

1. Write the trigger that fires when the db is modified, record the information in audit trail with trigger event.

```
Create or replace trigger t1
After delete or update
On client-master
For each row
Declare
Client no 1 client-master, client no % type;
Name, client-master, name % type;
Balance due1 client-master, balance due % type;
Oper varchar(20);
Begin if updating then oper:='update'
End-if: if deleting then oper:='deletion'
End-if:
        Client no1:= old.client no;
        Name1:= old.name;
        Balance due 1: old.balance due;
        Insert into audit trail(client-no1);
End;
```

```
1. Create trigger t1
    Before update of total on student
    For each row
    Declare
        Total 1
    Begin
        If new.total=0
            Raise application error(-20001,"don't enter
this"/);
        End if;
    end
```

FORMS

AIM: About forms

Tools Provided By Oracle Developer/2000

Oracle Developer/2000 provides four tools:

- Object Navigator: In this tool you can view all your objects, add new objects and name/rename your objects.
- Layout Editor: This tool helps you design your forms and reports and add various objects to them like push buttons and list boxes.
- PL/SQL Editor: This is the tool that is used to write all the codes for the triggers, procedures or functions.
- Menu Editor: This tool will help you create a customized menu that can be attached to your form or report.

Logging On to Developer/2000

1. To log on to Developer/2000, go to Start → Developer R2.1 and select Form Builder (See Figure 8.1)

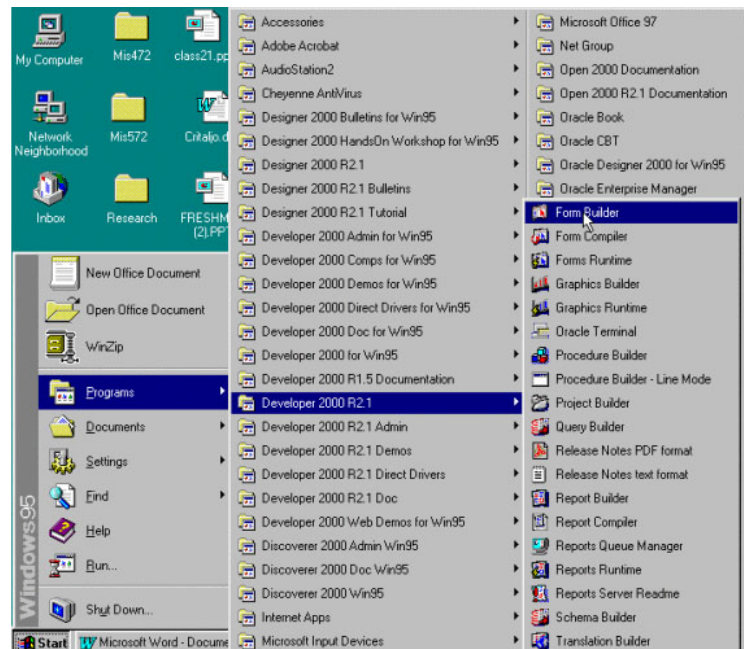


Figure 8.1: The Start Menu for Logging in to Developer/2000

2. Immediately, you will see the window for Developer/2000 Form Builder for Windows 95/NT with the sub-window for Welcome to Form Builder. (See Figure 8.2)

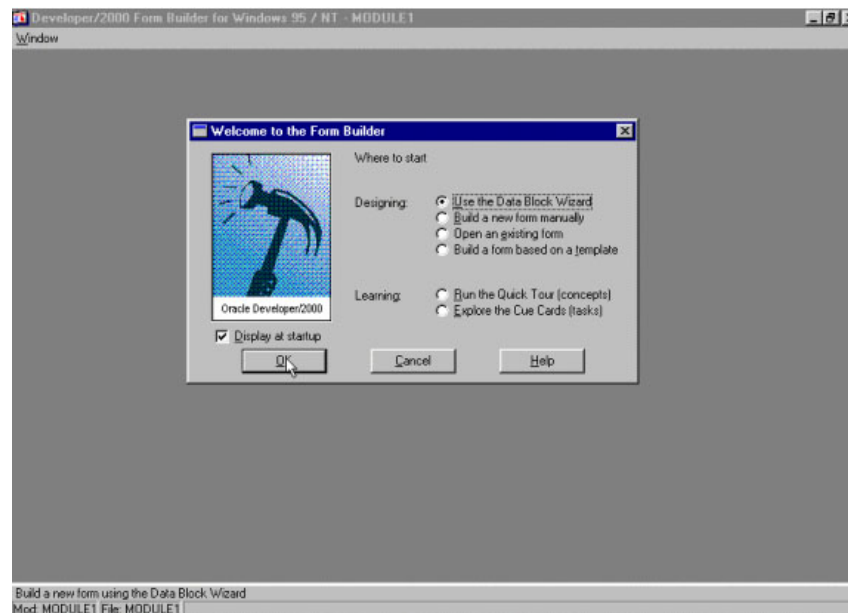


Figure 8.2: The Form Builder window

3. Select 'Use the Datablock Wizard' and click OK--This is the easiest method to design a new form.
4. You will now see the Welcome to the Datablock Wizard Window. Click Next to proceed. (See Figure 8.3)



Figure 8.3: Welcome to Datablock Wizard Window

5. You will now see the window for the Datablock Wizard. Select Table or View as in the figure and click Next. (See Figure 8.4)

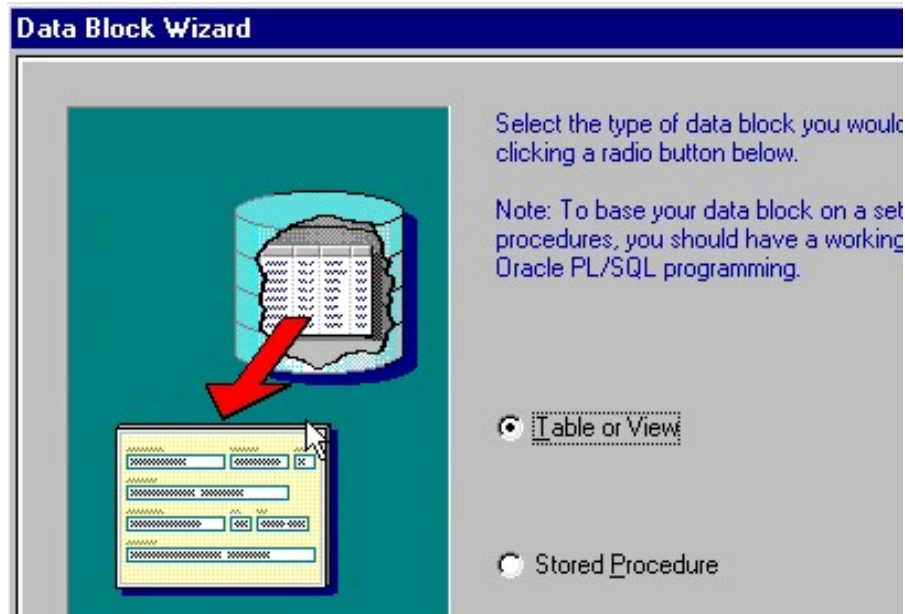


Figure 8.4: The Datablock Wizard Window

6. You will now see the window that prompts you to select a table or a view--your form will be created based on this selection. Since no table or view is being shown, click on browse to look at the list of tables and views in your database. (See Figure 8.5)

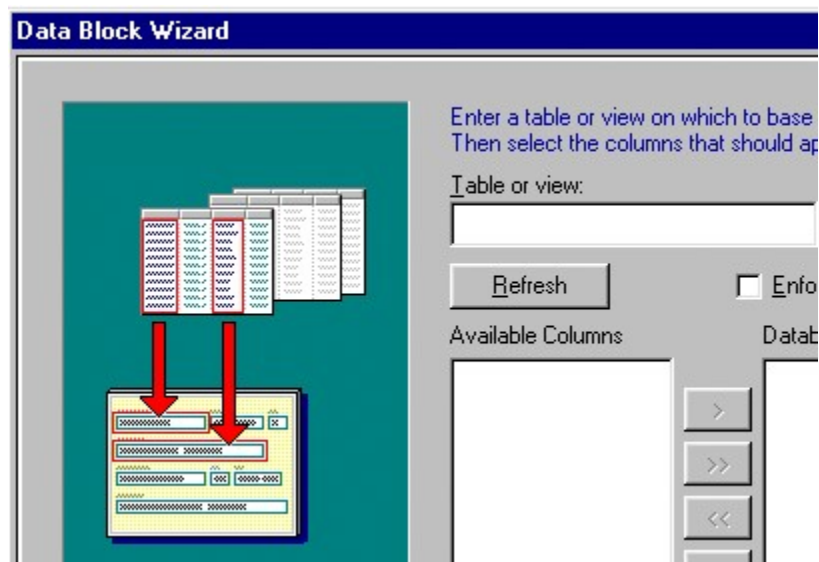


Figure 8.5: The window for selecting the base table

- Once you click browse, the connect window will appear. Type in your username, password and database to connect to the database. (See Figure 8.6)

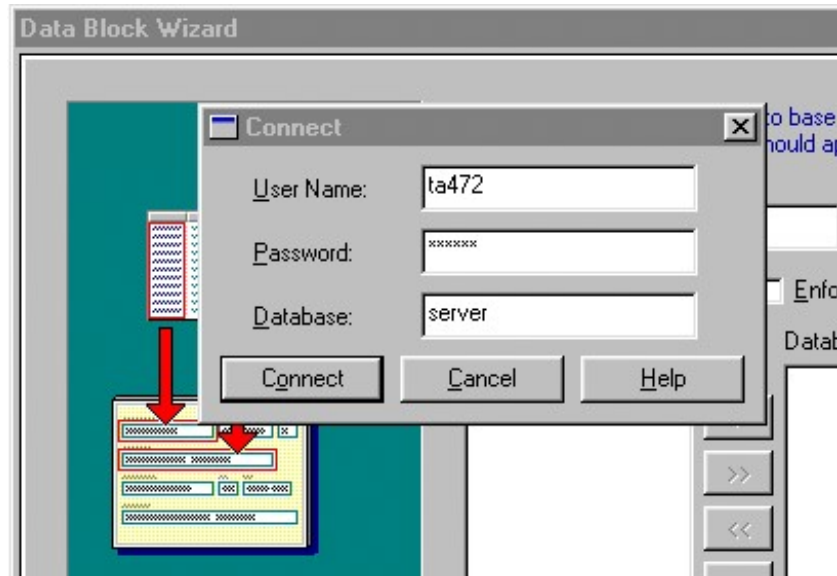


Figure 8.6: The Connect String Window

(We typed "server" in the Database field because our data resides on a central server. If your data resides on your personal hard drive, this field can be blank.)

- You will now see the tables window. Select current users and tables and click OK. (See Figure 8.7)

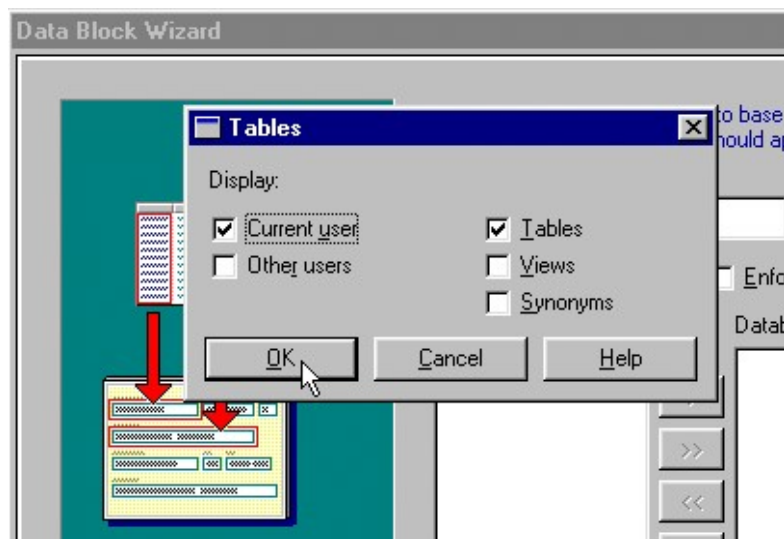


Figure 8.7: The Tables Window

9. You will now see the list of tables created in your database. Select Students and click OK. (See Figure 8.8)

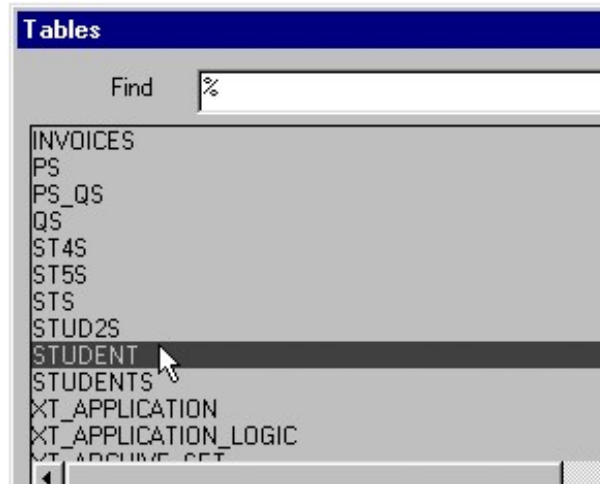


Figure 8.8: The window with the list of tables in the database

10. You will now see your selected table and its available columns on your screen. Click on the single right arrow to select the first column to be shown in your form; in this case the STUDID column. You will now see this column under the database items selected sub-window. (See Figure 8.9)

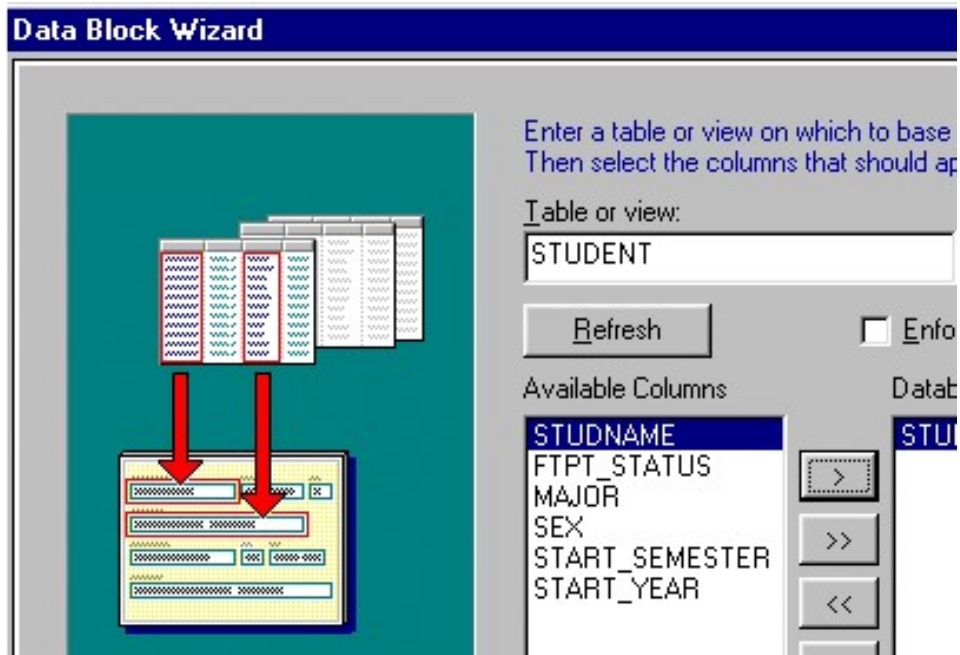


Figure 8.9: The window with the selected table and its available columns.

11. To move the rest of the columns, simply click on the double right arrow and this will select all your columns in to the database items. (See Figure 8.10)

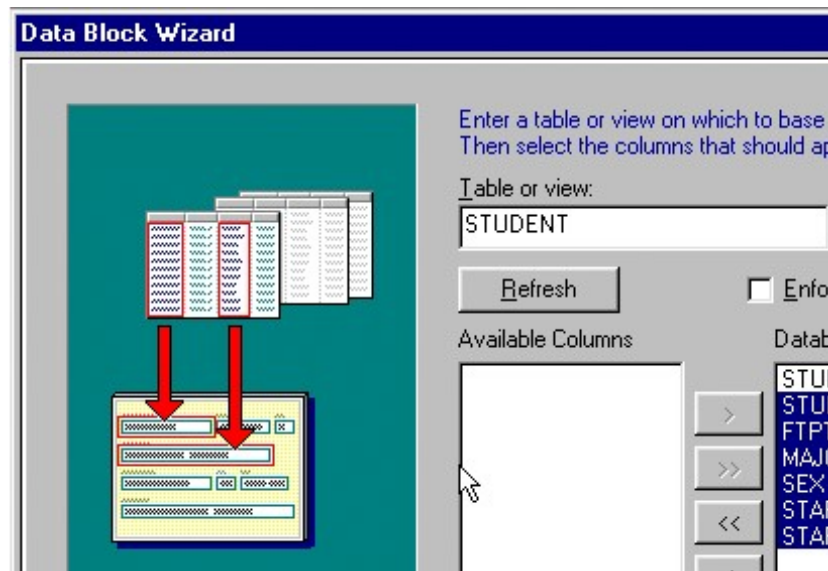


Figure 8.10: The Database Wizard Window with all the selected columns of the base table

12. You will now see the Congratulations window. Make sure that "Create the data block, then call the Layout Wizard" is selected and click on Finish. (See Figure 8.11)



Figure 8.11: The Data Block Wizard Congratulations Window

13. You will now see the Layout Wizard, prompting you to select the items that you would like to show in the form. Make sure that the data block selected is Students and then click the double right arrow to move all the columns of the Student block from the available items to the displayed items. Click on Next to continue. (See Figure 8.12)

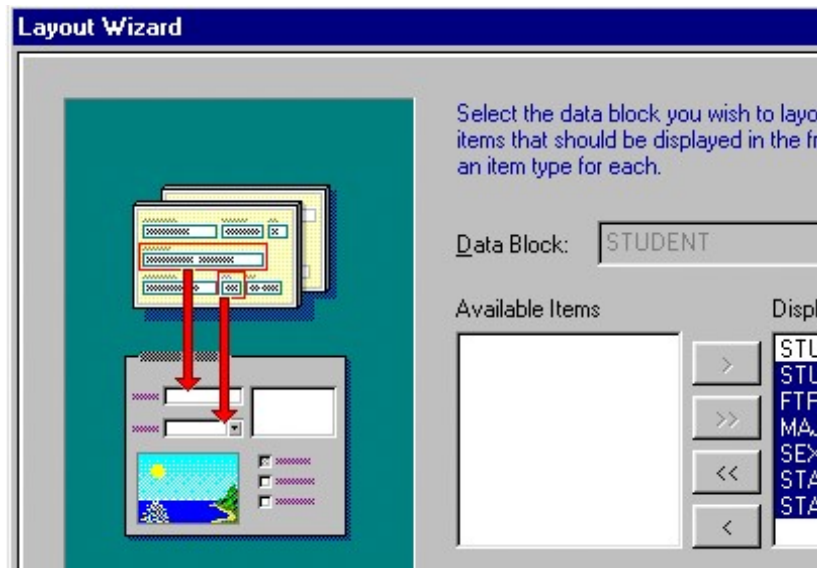


Figure 8.12: The Layout Wizard Window

14. The window with the prompt for the height and width of the items will appear. Click Next to accept the default values. (See Figure 8.13)

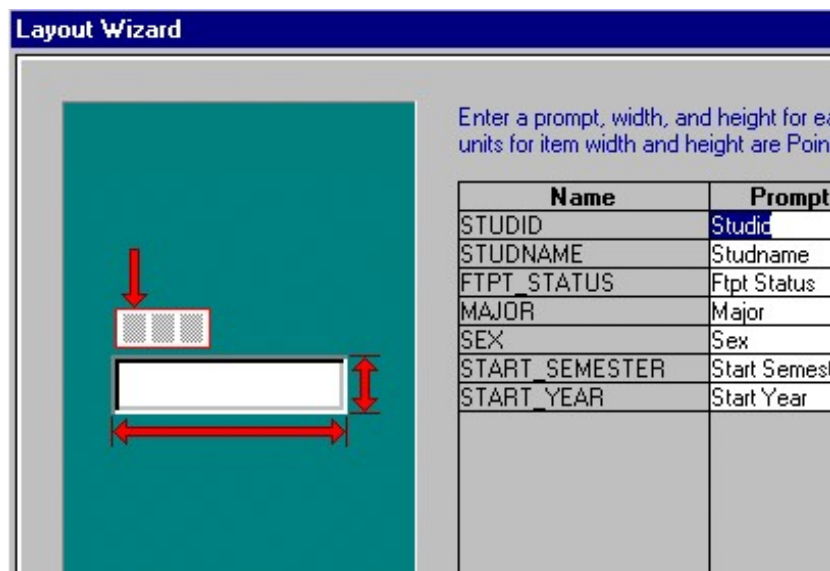


Figure 8.13: The window displaying the height and width of the items selected

15. The Layout Wizard will now prompt you to select the layout or view style of your block. Select Form and click Next. (See Figure 8.14)

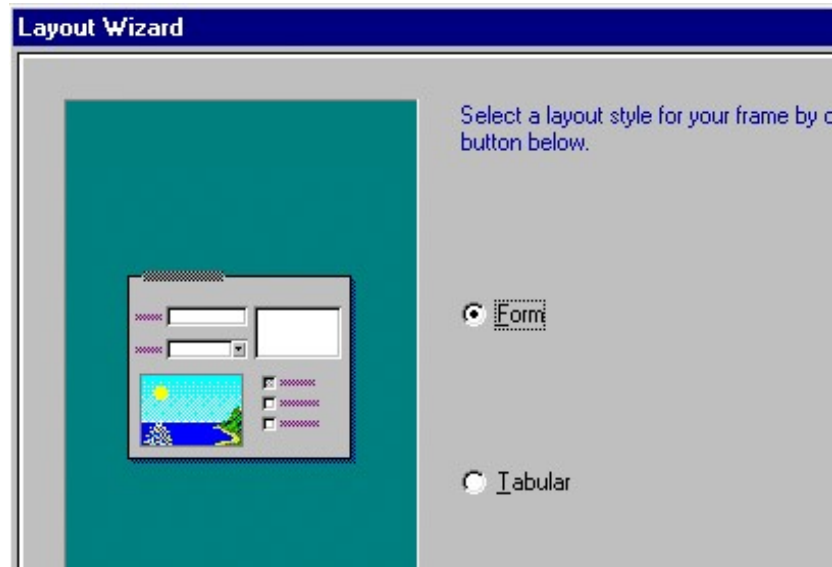


Figure 8.14: The window for selecting the layout style of the selected table

16. The Layout Wizard will now prompt you to select a title for the form that you are creating. Type in Student Records. Click Next to continue. (See Figure 8.15)

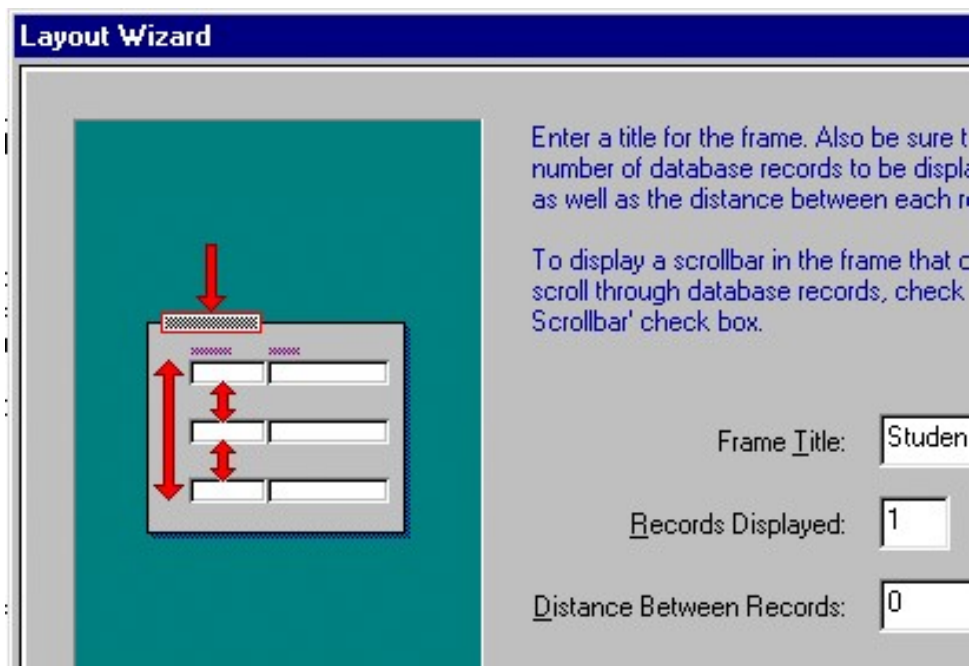


Figure 8.15: The Window for selecting the appropriate title for the Form

17. Congratulations! You have now successfully created your first form. Click Finish to view your form. (See Figure 8.16)



Figure 8.16: The Layout Wizard Congratulations Window

18. You will now see the canvas view of the form that you have created. You can now add various objects like push buttons, combo boxes and radio buttons to your form to make it more graphical and user friendly. We will do this in the next lesson. (See Figure 8.17)

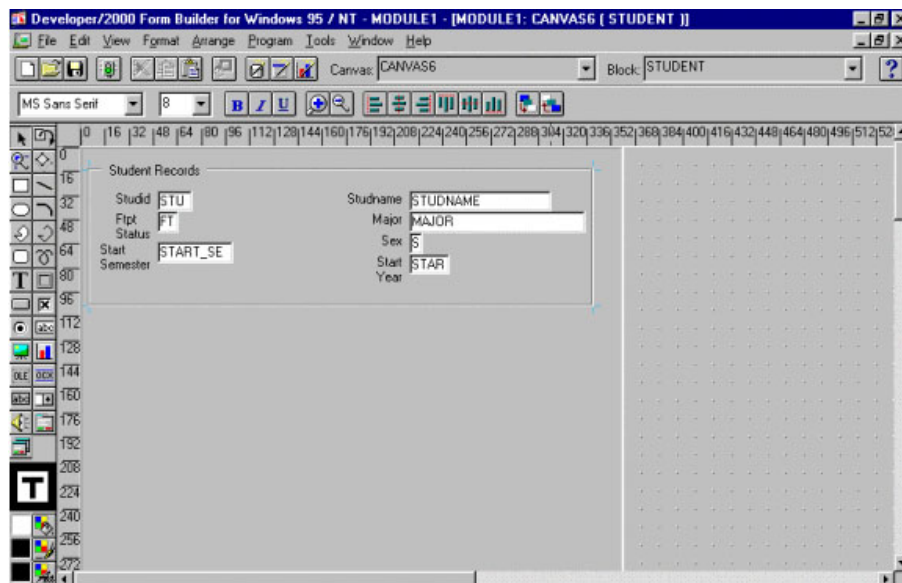


Figure 8.17: The canvas view of the newly created form

19. You can now format the form manually. Click on the frame to select it. Then drag the frame to make it bigger. (See Figure 8.18)

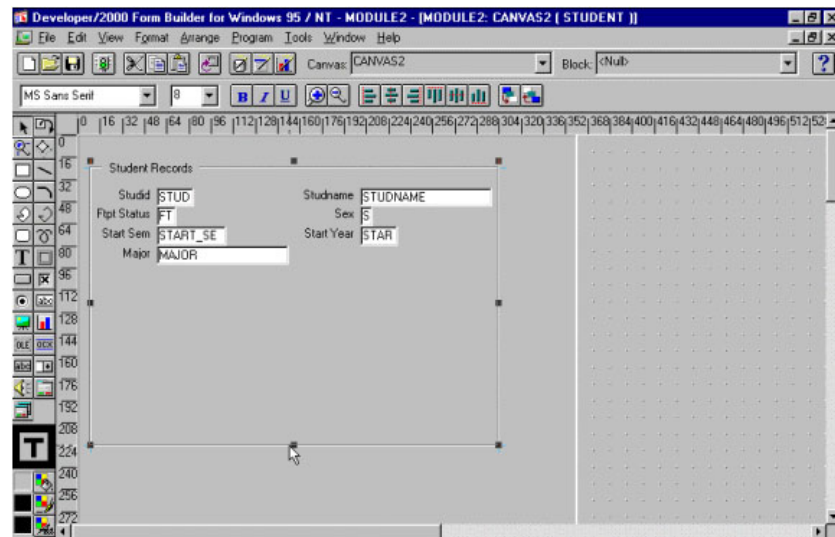


Figure 8.18: Formatting the size of the frame

20. You can now space out the data fields to make your form more visually appealing. You can do this by simply selecting the data field and dragging it to your desired area. (See Figure 8.19)

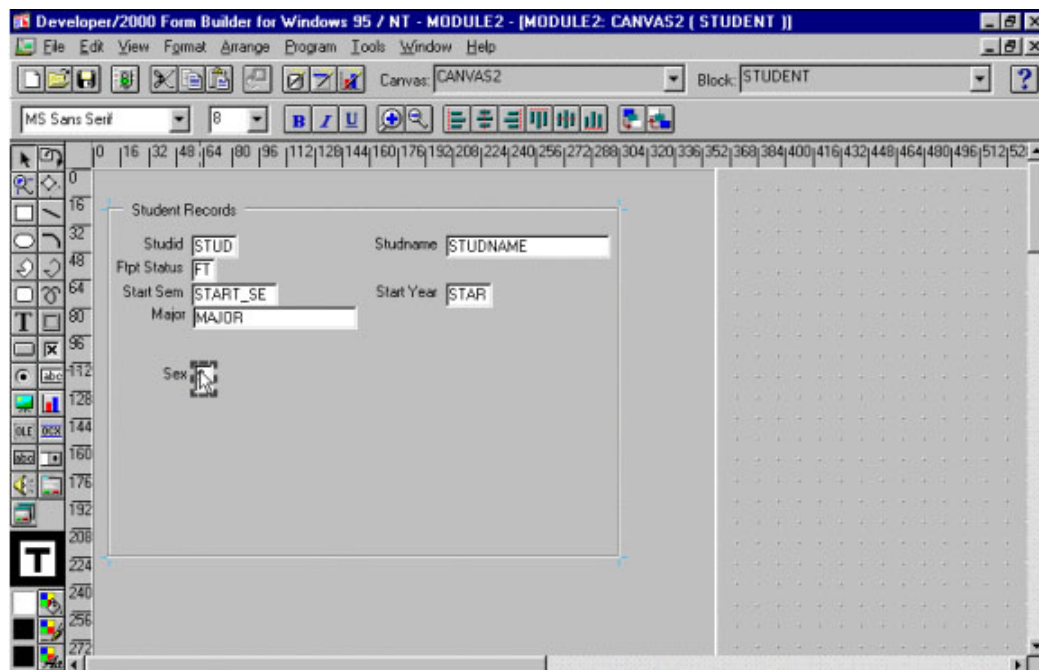


Figure 8.19: Spacing out the data fields

21. After you have formatted all the data fields, your form should look like Figure 8.20.

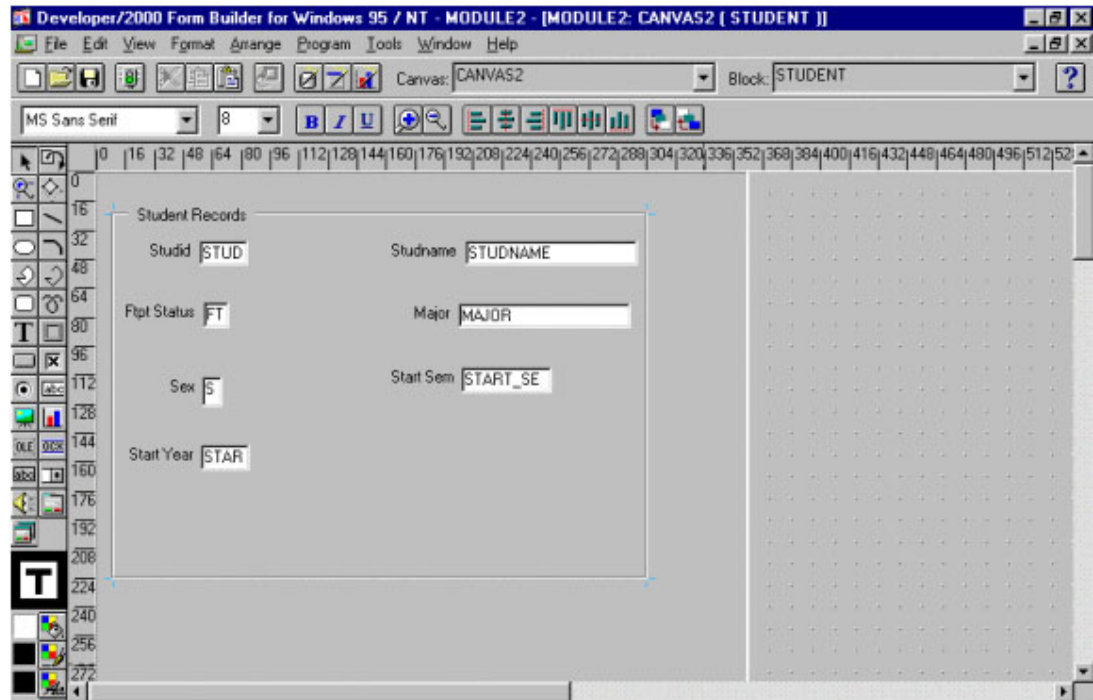
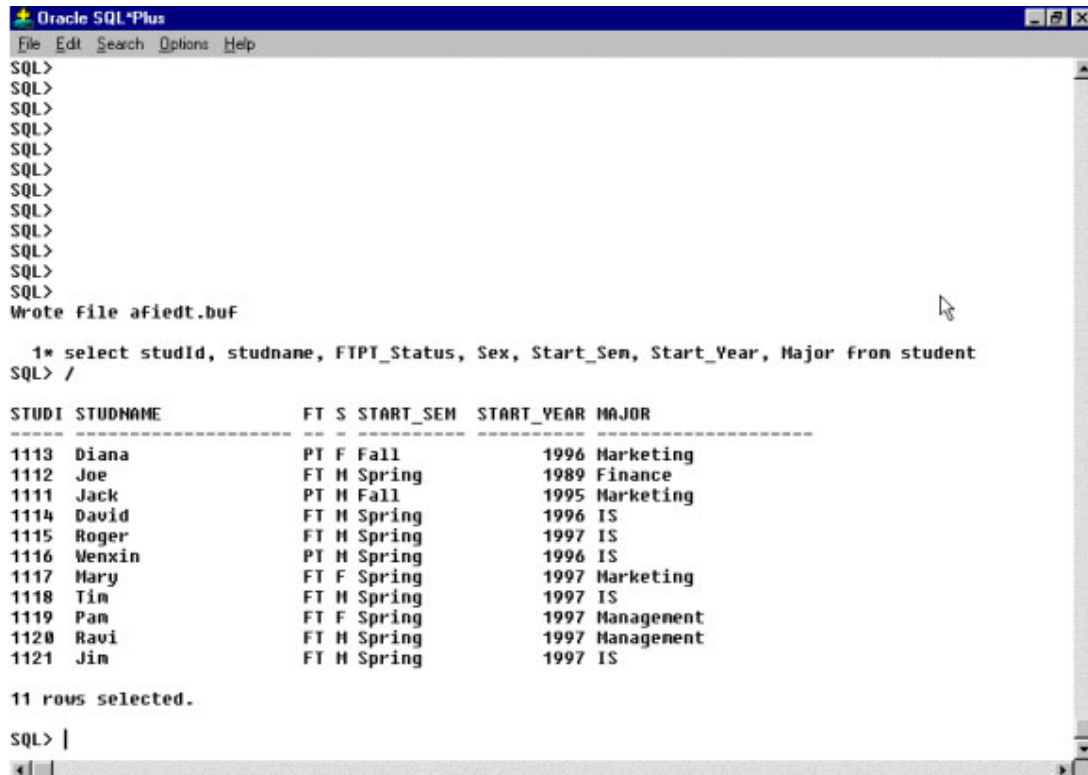


Figure 8.20: The formatted form

REPORTS

AIM: About Reports

Our objective will be to create a simple report that will list students along with some student attributes. Students in this report will be categorized by major. As a starting point, let us assume that the STUDENT table has the following rows. (See Figure 11.1).



```
Oracle SQL*Plus
File Edit Search Options Help
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
Wrote file afiedt.buf

1* select studId, studname, FPT_Status, Sex, Start_Sem, Start_Year, Major from student
SQL> /
```

STUDI	STUDNAME	FT	S	START_SEM	START_YEAR	MAJOR
1113	Diana	PT	F	Fall	1996	Marketing
1112	Joe	FT	H	Spring	1989	Finance
1111	Jack	PT	H	Fall	1995	Marketing
1114	David	FT	H	Spring	1996	IS
1115	Roger	FT	H	Spring	1997	IS
1116	Wenxin	PT	H	Spring	1996	IS
1117	Mary	FT	F	Spring	1997	Marketing
1118	Tim	FT	H	Spring	1997	IS
1119	Pam	FT	F	Spring	1997	Management
1120	Ravi	FT	H	Spring	1997	Management
1121	Jim	FT	H	Spring	1997	IS

```
11 rows selected.
SQL> |
```

Figure 11.1: The Student table as viewed in SQL *Plus

You may enter the data shown below using the INSERT statement in SQL*Plus covered earlier in the tutorial (see Lesson 9). For example, the following statement may be used to insert the first row (for the student Diana with Student Id '1113').

```
SQL> insert into student values ('1113', 'Diana', 'PT', 'F', 'Fall', 1996, 'Marketing', 'Y');
```

Enter the rest of the data as shown in Figure 11.1.

Creating Reports:

1. In order to create reports, you will need to go to the Reports Builder in Oracle Developer/2000. To do this, go to the Start button and select Programs→Developer 2000 R2.1→Report Builder (See Figure 11.2)

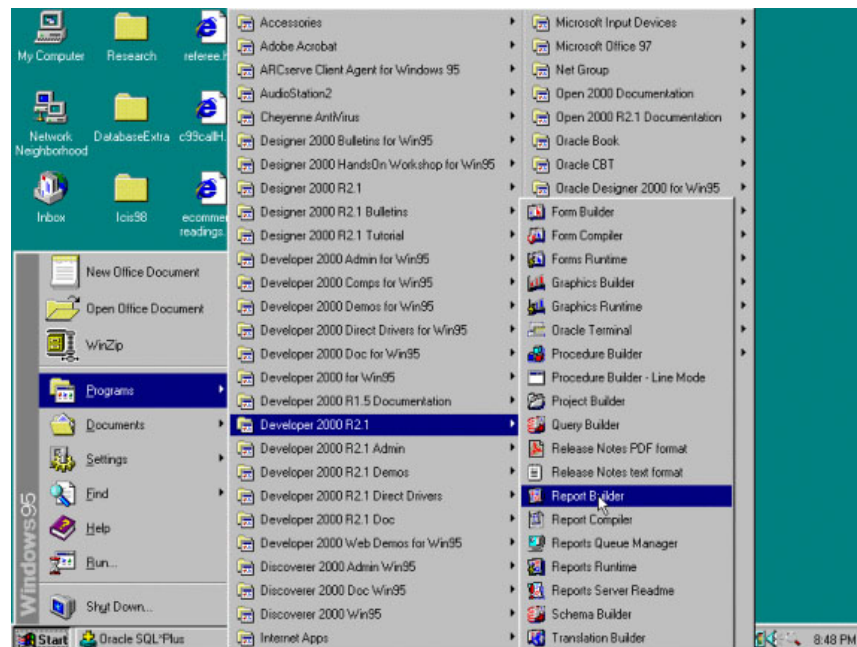


Figure 11.2: Logging on to Report Builder

2. Immediately, you will see the Welcome to Report Builder Window. Select the radio button for the *Build a new report manually* and click OK. (See Figure 11.3)

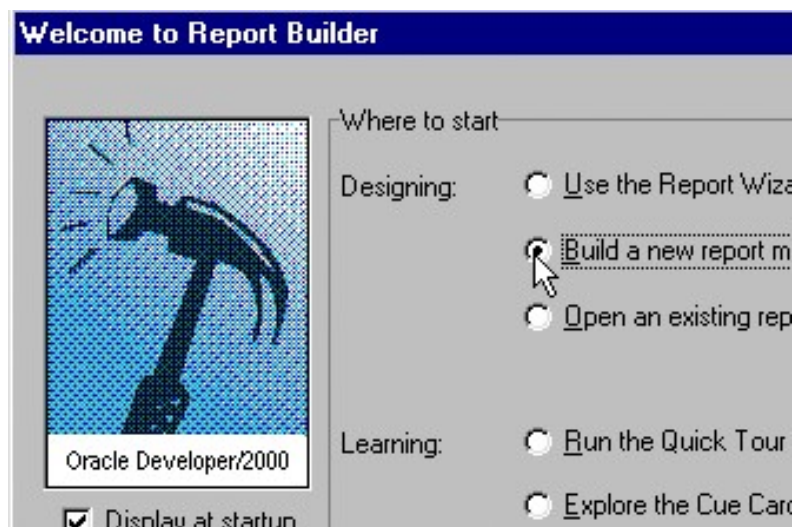


Figure 11.3: The Welcome to Report Builder window

3. Once you click OK, you will see the Report Editor-Data Model window, with a default name for the Data Model. (See Figure 11.4)

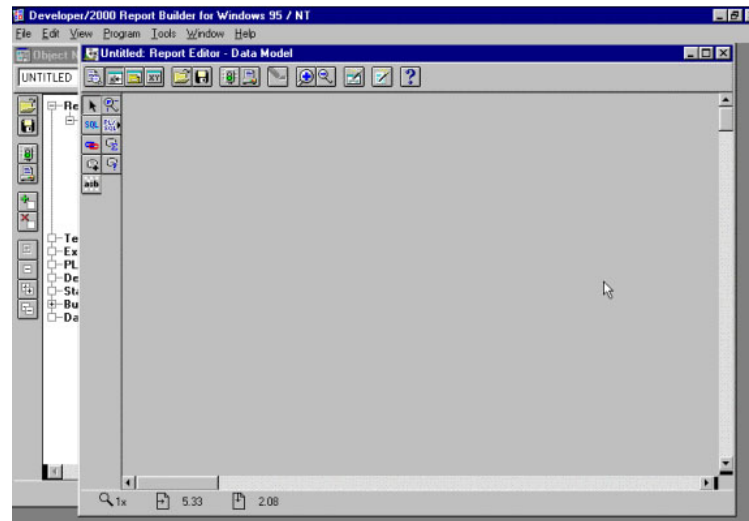


Figure 11.4: Window for the Report Editor- Data Model

4. The Report Editor is the tool that you will use to create your data model for the report. Click on the SQL icon (See Fig. 11.5) in the toolbar located on the left hand side, and drag and drop it on the palette. Immediately, the SQL Query Statement window will appear. Type in the displayed SQL query to view student information in order of major. (See Figure 11.6)



Figure 11.5: The SQL icon in the toolbar of the Report Editor

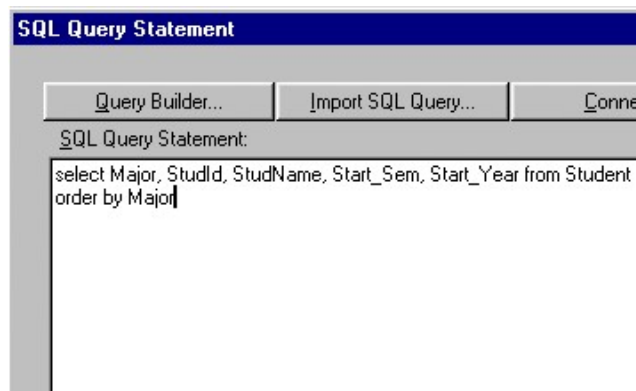


Figure 11.6: The SQL Query Statement window

5. Once you click OK, Developer/2000 will prompt you to connect to the database. Type in your User Name, Password and Database. (See Figure 11.7).



Figure 11.7: The Connect window for connecting to the Database

6. You will now see your data model, where Q_1 stands for the name of the query and G_Major, stands for its associated record group which contains the list of fields that you will be able to view in your report. (See Figure 11.8)

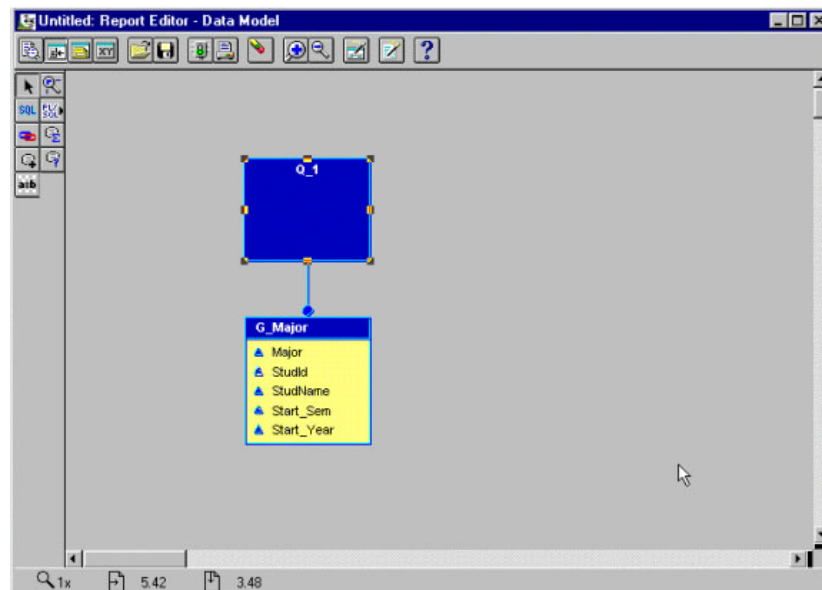


Figure 11.8: The Report Editor with the Data Model

7. To change the name of your query, right click on it and select the Property Palette. (See Figure 11.9)

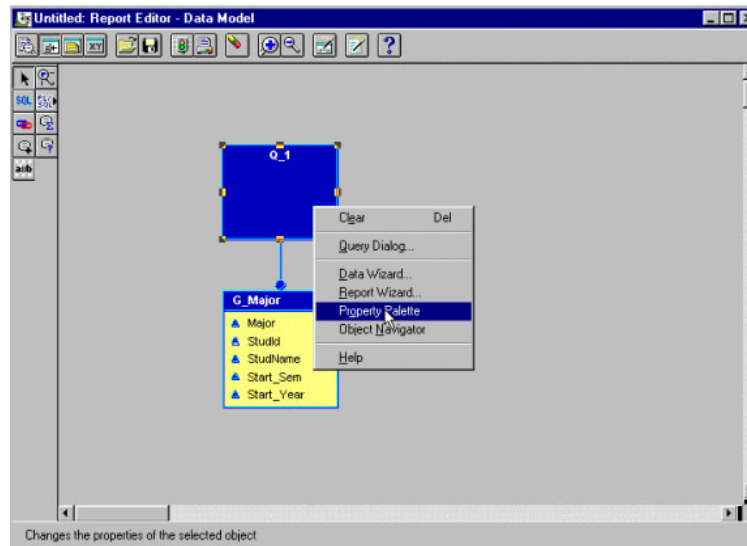


Figure 11.9: Selecting the Property Palette of the Query

8. Immediately, you will see the window for the Property Palette. Change the name by typing in the name (Q_StudMajor) beside the 'Name' tab, and press enter. You can also change or edit your SQL query by clicking on the space beside the SQL Query Statement tab. (See Figure 11.10)

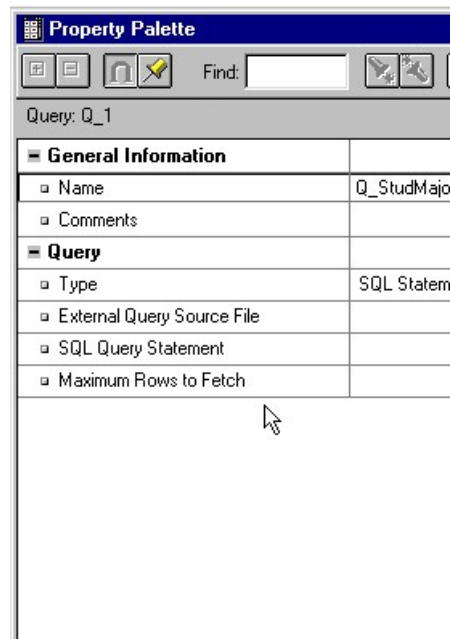


Figure 11.10: The Property Palette window for the Query

9. Your Data Model should now look like the one in Figure 11.11.

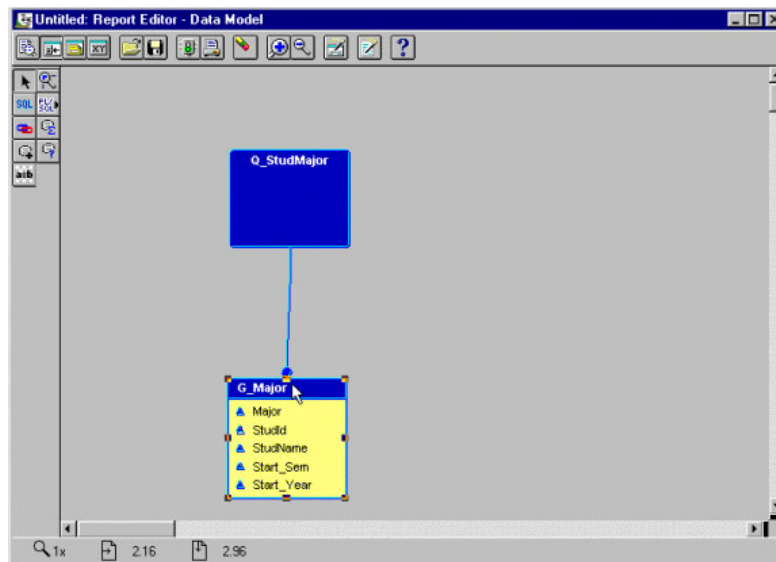


Fig. 11.11: The Edited Data Model

10. Recall that we have been asked to create a report that will display a list of students and their related information organized by Majors. To do this, we will move the Major records into a separate record group. In Oracle Developer/2000 terms, it is called to 'break out'. First, click on the G_Major, and drag and lower it to create more space between the record group and the query. Then select Major, and drag and drop it on the line connecting Q_StudMajor and G_Major. (See Figure 11.12)

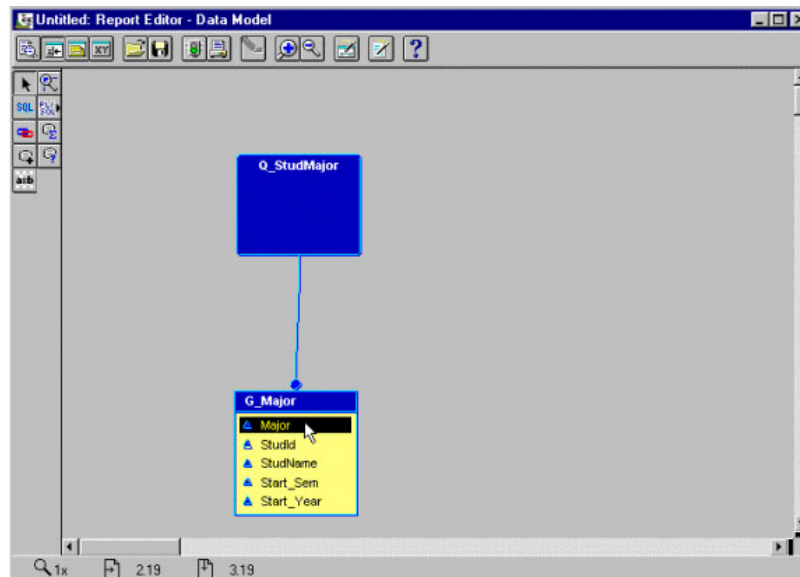


Figure 11.12: Creating a Break Group

11. Your Data Model should now look like the one in Figure 11.13 with a new group for Major.

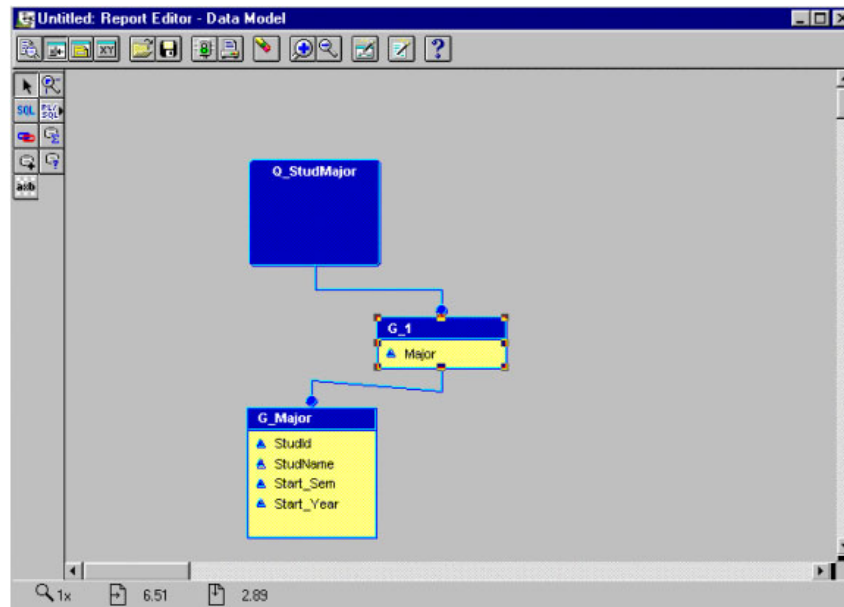


Figure 11.13: The Edited Data Model with a separate record group for Major

12. Right click on the *G_Major* to go to its Property Palette. Change its name to *G_Break*. (See Figure 11.14)

Figure 11.14: The Property Palette for the Major record group

13. Your Data Model should now look like Figure 11.15.

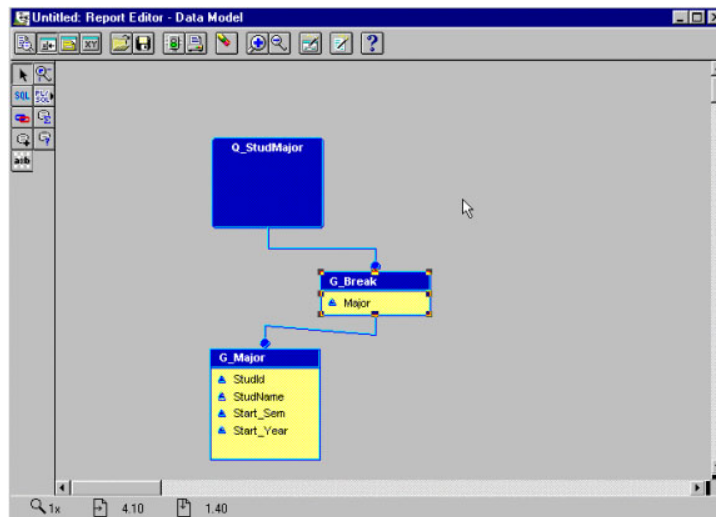


Figure 11.15: The Updated Data Model

14. Now select Report Wizard from the Tools Menu to generate the report.
(See Figure 11.16)

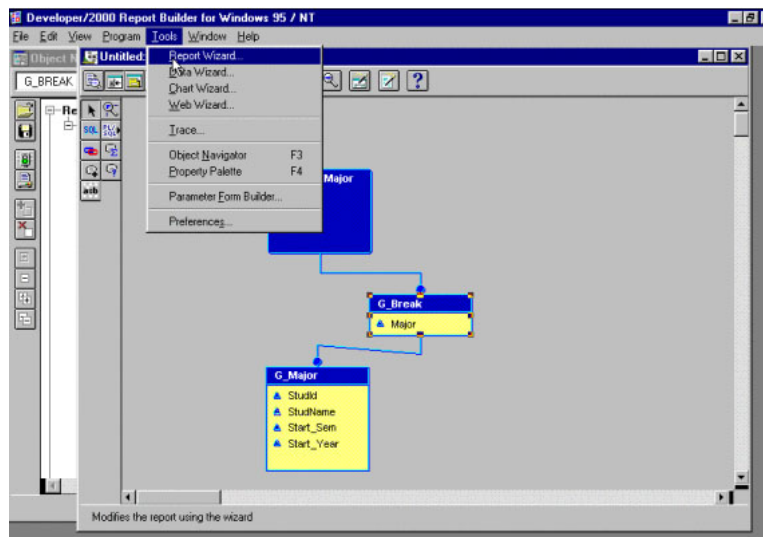


Figure 11.16: Selecting the Report Wizard from the Tools Menu

15. You will now see the first screen of the Report Wizard. Type in "List of Students by Major" in the Title box. Next, select the radio button for *Group Above* in order to create breaks after record groups for each Major. Now, click Next. (See Figure 11.17)

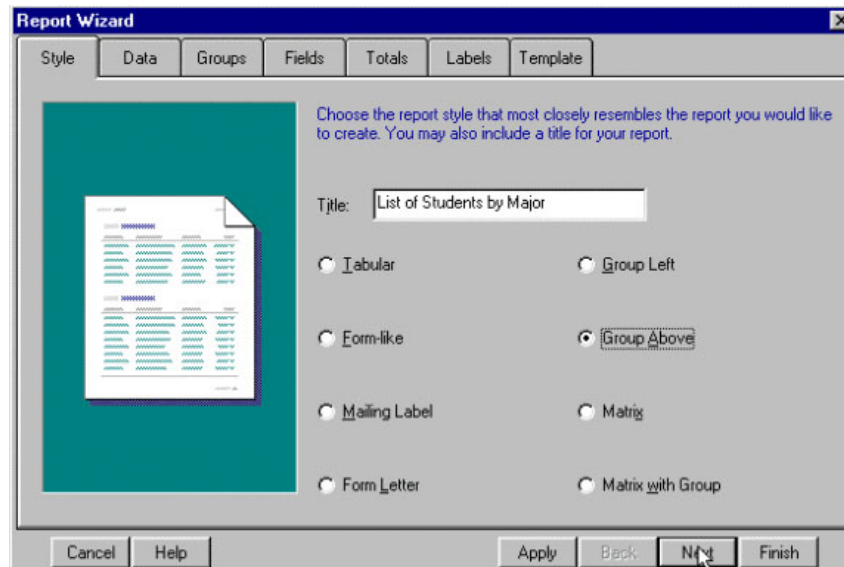


Figure 11.17: Selecting a report style using the Report Wizard

16. You will now see your SQL statement again. You can edit your statement here if you choose to. At this time we will use the query that we had entered earlier. Click Next. (See Figure 11.18)

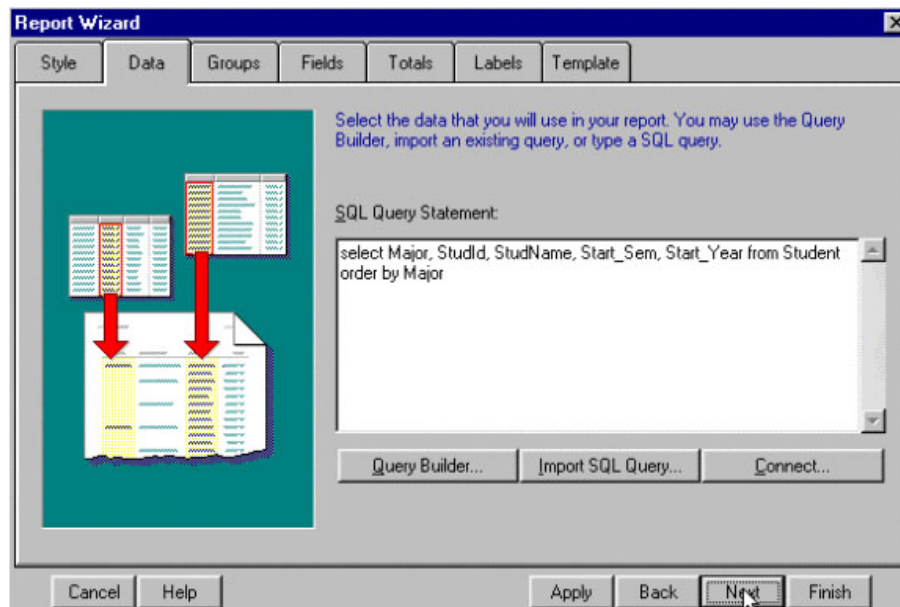


Figure 11.18: Editing the SQL Statement

17. You will now be prompted to select the fields that you would like to designate as group fields. You will find that Oracle has already selected Major. (See Figure 11.19). Now, select the next tab, Fields.

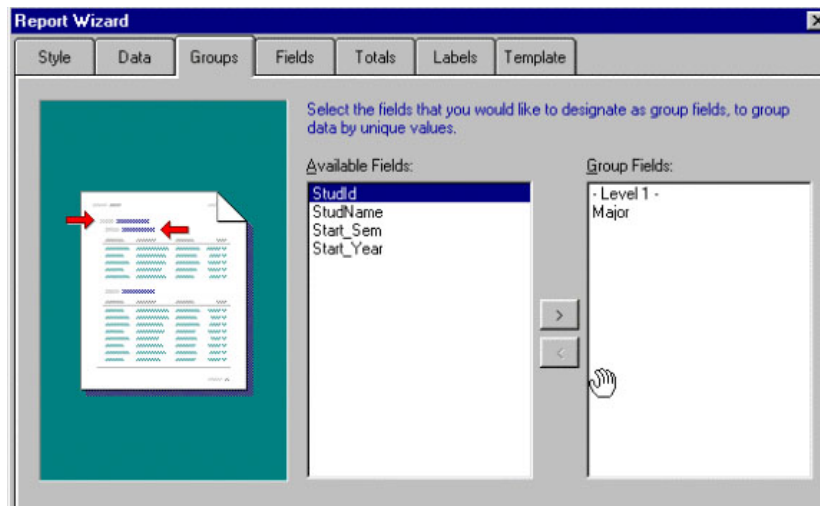


Figure 11.19: Selecting a field to Designate as group Field

18. You will now be asked to select the fields that you will display in your report. We would like to see all the fields, so click on the double right facing arrows to select all the fields and click Next. (See Figure 11.20).

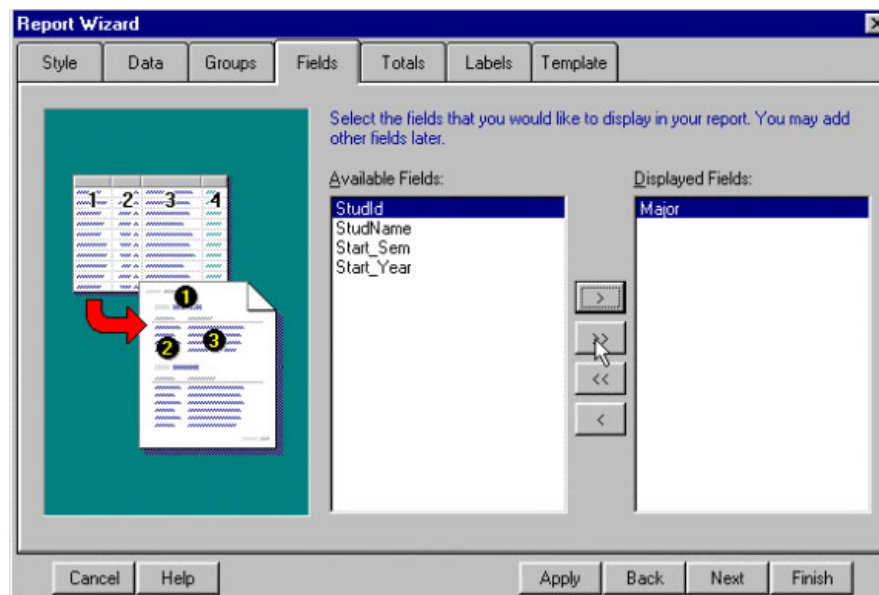


Figure 11.20: Selecting the Fields that are to be displayed in the Report

19. You will now be prompted to select fields for which you would like to calculate totals. Let us assume that we have been asked to provide the total number of students in each major and also the Grand total of the number of students. To do this, select StudID, and click on Count. (See Figure 11.21).

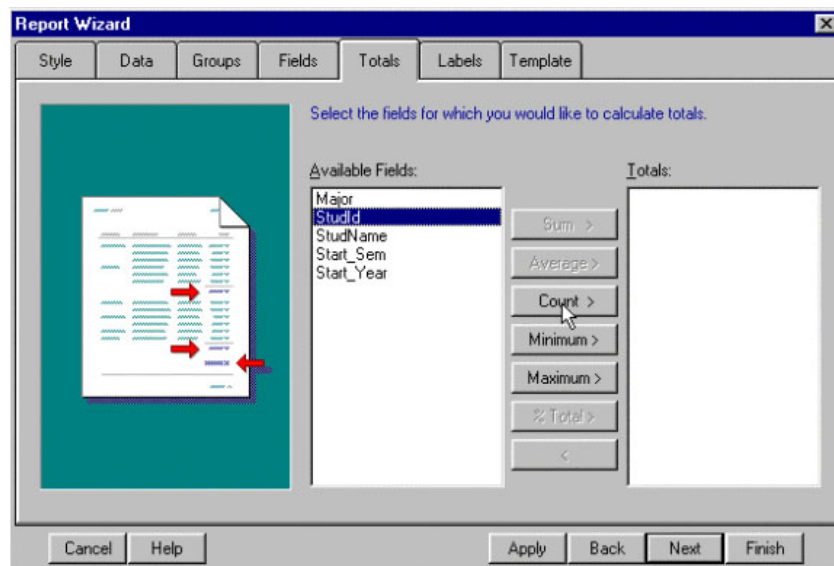


Figure 11.21: Selecting a Field on which you will calculate a total

20. Your Screen should look like Figure 11.22 with Count (StudID) in the Totals column). Click Next.

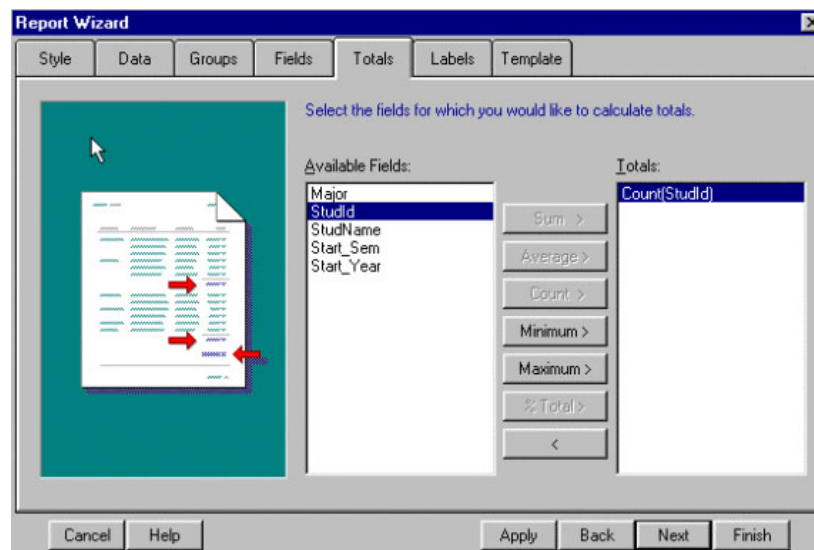


Figure 11.22: A view of the Totals Screen

21. You can now modify your labels and their width. In this case we have put a colon and a space after Major and have changed the label for CountStudIdPerReport to "Number of Students: " and click Next. (See Figure 11.23)

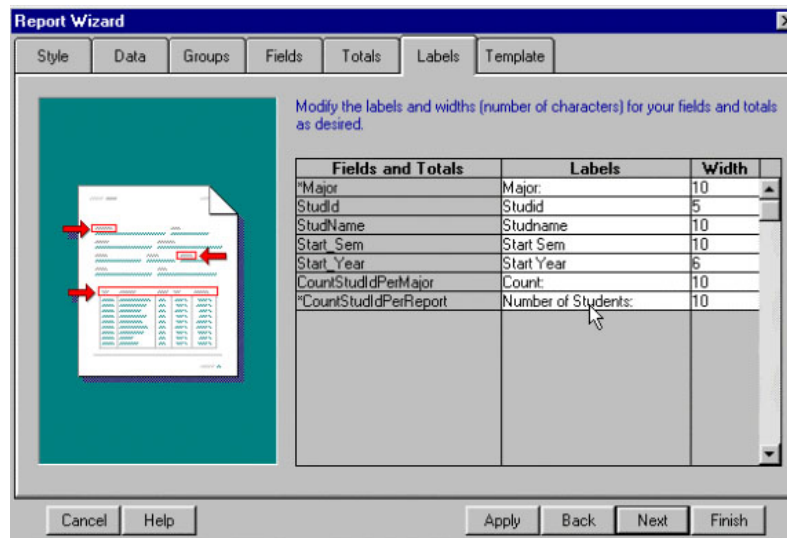


Figure 11.23: Modifying the labels and their width of the report.

22. The final modification involves selecting an appropriate template for the report. In this case, we will select Corporate2 from the list provided. You are free to select any template of your choice. Click Finish. (See Figure 11.24)

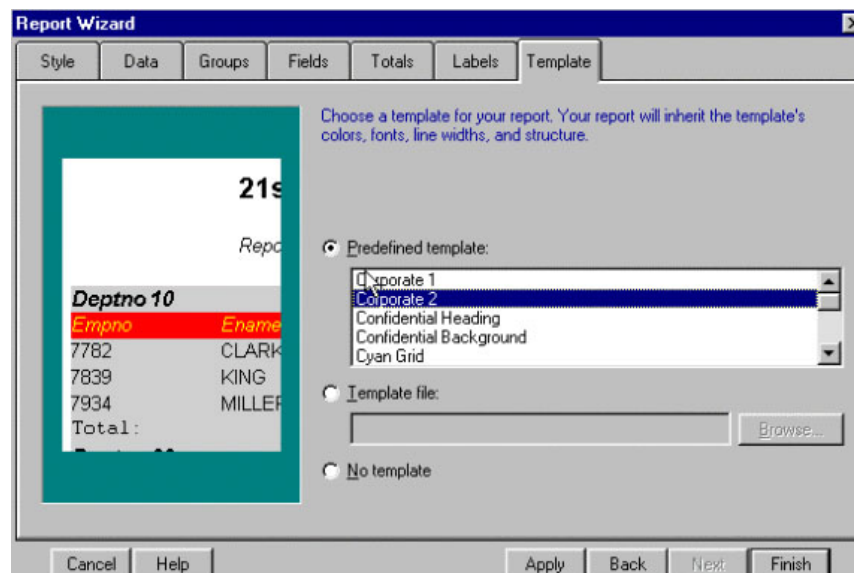


Figure 11.24: Selecting an appropriate template for the report

23. Your report should now look like the one in Figure 11.25.

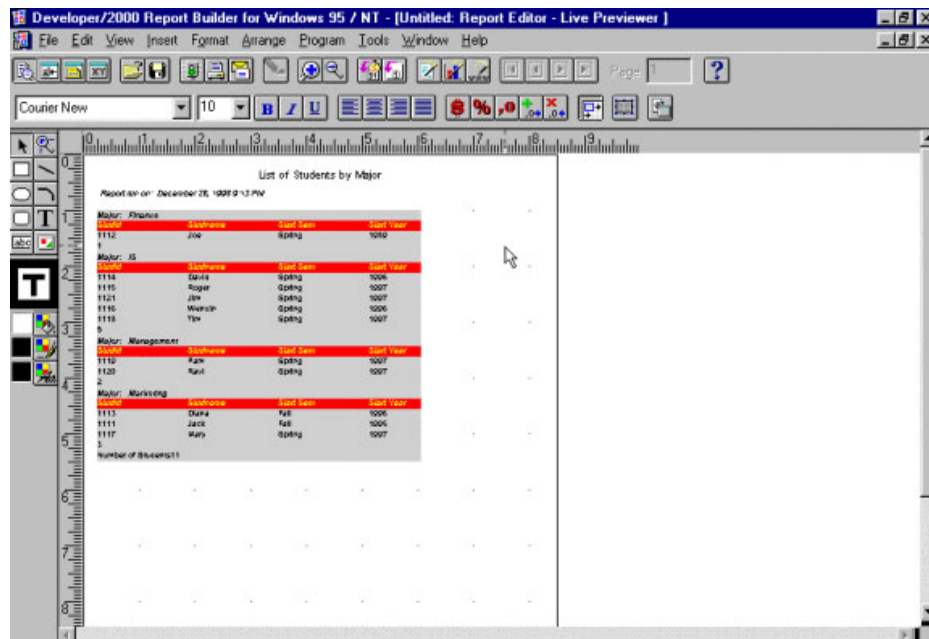


Figure 11.25: The newly created report

24. The following figure provides a closer look at the created report. We will now create a chart that will let us compare the distribution of students in each major. For this, select the Chart Wizard by clicking on the icon with the tiny bar graphs as indicated by the arrow in Figure 11.26.

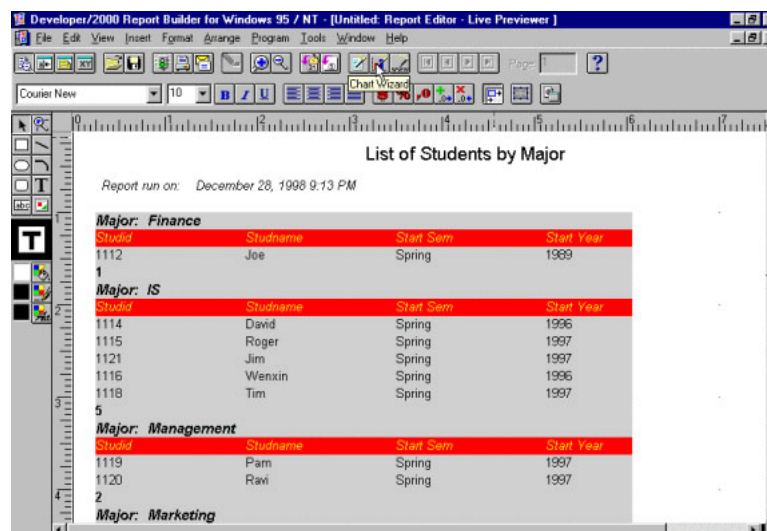


Figure 11.26: Selecting the Chart Wizard

25. You should see the *Welcome to the Chart Wizard* window. Click Next.
(See Figure 11.27)

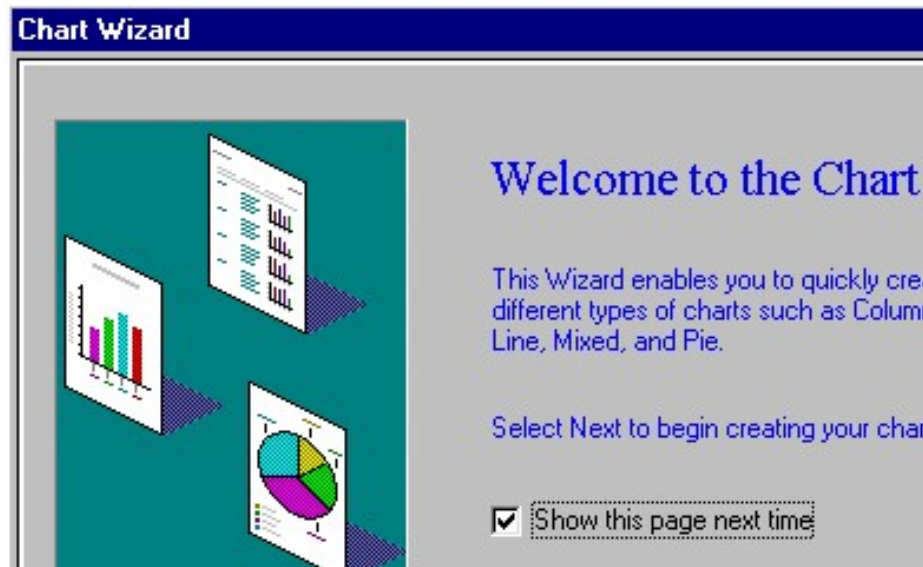


Figure 11.27: The Welcome to the Chart Wizard Window

26. You will now be prompted to enter a chart title, select a chart type and a chart subtype. Type in "Distribution of Students by Majors" in the box for the title. Select Pie as the type of chart and Depth as the Chart Subtype. (See Figure 11.28)

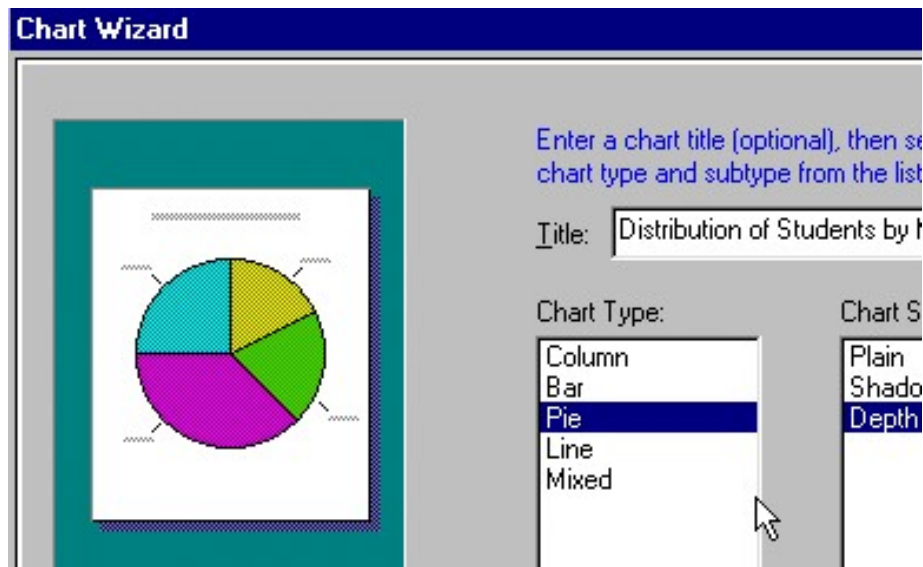


Figure 11.28: Selecting a Chart type and subtype and providing a title for the Chart

27. You will now have to select a field for the X-axis. Select Major and click on the right arrow. (See Figure 11.29)

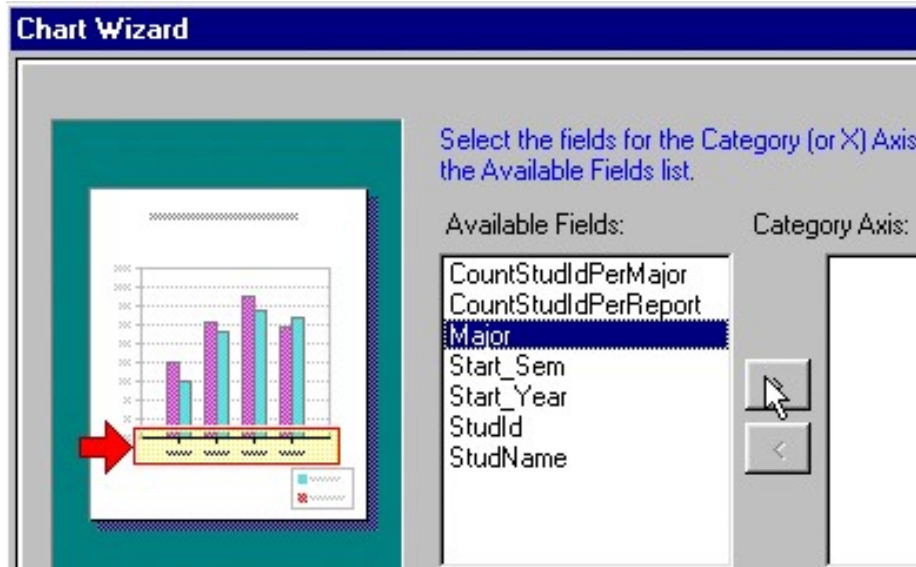


Figure 11.29: Selecting a field for the X axis

28. Now select CountStudIdPerMajor as the field for the Y-axis and click on the right arrow. (See Figure 11.30)

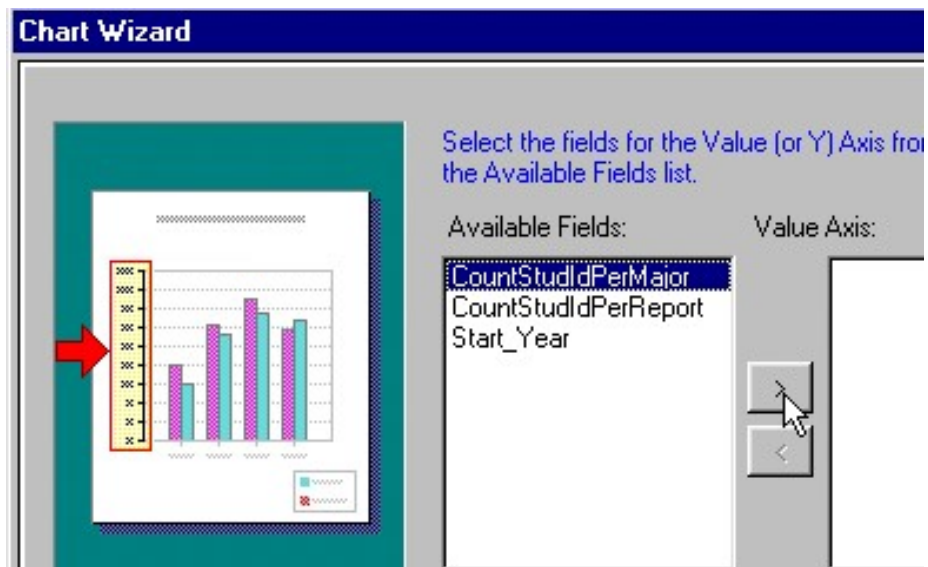


Figure 11.30: Selecting a field for the Y axis

29. You will now have to select a chart placement for the chart in the report. Let us assume that we would like the chart to be placed at the end of the report. Select *at the end of the report* and click Next. (See Figure 11.31)

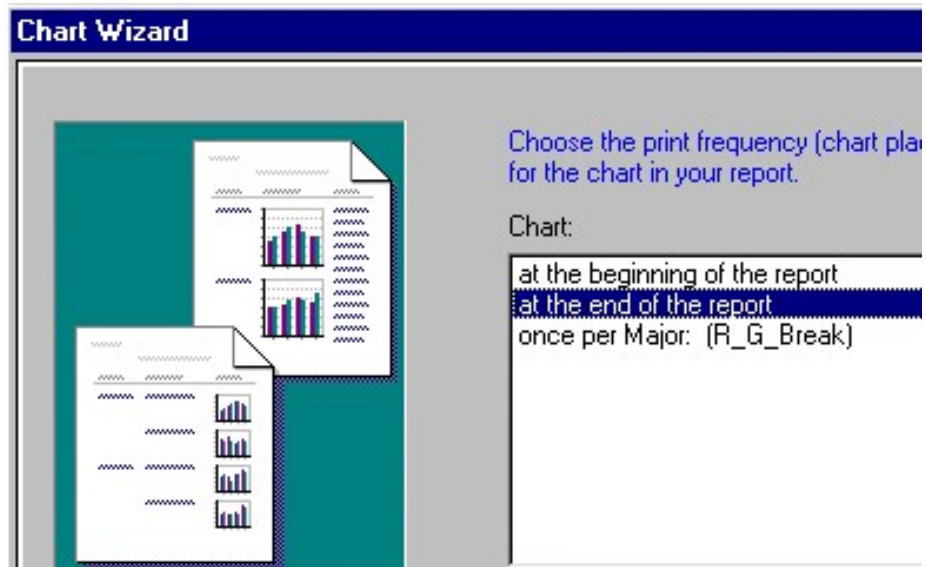


Figure 11.31: Selecting an appropriate chart placement for the chart in the report

30. You will now see the Congratulations window. Click on Next. (See Figure 11.32)



Figure 11.32: The Congratulations Window for successfully creating the chart

31. Your chart should now look like Figure 11.33.

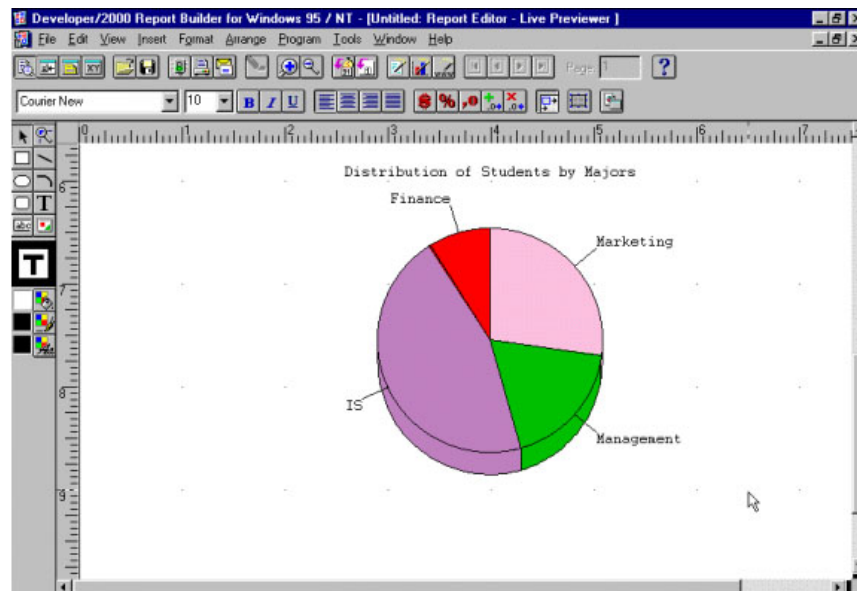


Figure 11.33: The newly created chart

32. See Figure 11.34 to get a view of your report with the chart at the end of it.

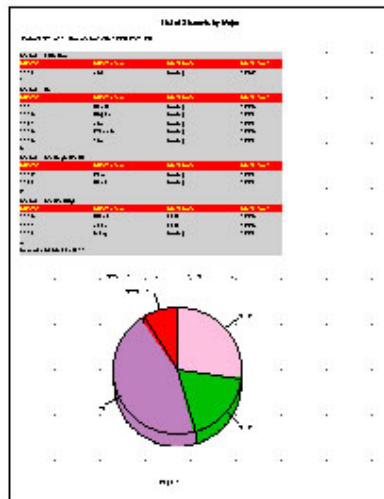


Figure 11.34: The report with the chart at the end of it

33. You can now save your report by selecting the diskette icon from the toolbar of the report or by selecting File → Save as SRSrpt. Oracle will save your report in the Bin folder under Orawin 95 as a .rdf.
34. You can also print your report by selecting File → Print.

ASSERTIONS

An assertion is a predicate expressing a condition we wish the database to always satisfy.

Domain constraints, functional dependency and referential integrity are special forms of assertion.

Where a constraint cannot be expressed in these forms, we use an assertion, e.g.

Ensuring the sum of loan amounts for each branch is less than the sum of all account balances at the branch.

Ensuring every loan customer keeps a minimum of \$1000 in an account.

An assertion in DQL-92 takes the form,

```
create assertion assertion-name check predicate
```

Two assertions mentioned above can be written as follows.

Ensuring the sum of loan amounts for each branch is less than the sum of all account balances at the branch.

```
create assertion sum-constraint check
```

```
(not exists (select * from branch
```

```
where (select sum)amount) from loan
```

```
where (loan.bname = branch.bname >=
```

```
(select sum)amount) from account
```

```
where(account.bname=
```

```
branch.bname)))
```

Ensuring every loan customer keeps a minimum of \$1000 in an account.

```
create assertion balance-constraint check
    (not exists (select * from loan L
        (where not exists (select * from borrower B, depositor D,
account A
            where L.loan# = B.loan# and B.cname = D.cname
and
                D.account# = A.account# and A.balance >= 1000
            )))
```

When an assertion is created, the system tests it for validity.
If the assertion is valid, any further modification to the database is allowed only if it does not cause that assertion to be violated.

This testing may result in significant overhead if the assertions are complex. Because of this, the assert should be used with great care.

Some system developer omits support for general assertions or provides specialized form of assertions that are easier to test.