Savitribai Phule Pune University

F. Y. B. C. A. (Science) Semester-II

Lab Course – I & II

WorkBook

Name: ____

CollegeName:______________________________________

RollNo.:_____________________ Division: ___________________________

AcademicYear:__________________________________

Savitribai Phule Pune University

Section-I

F. Y. B. C. A. (Science)

SEMESTER II

BCA - 205

Lab Course - I
Advance C Assignments

Editors:

Section-I:

1. **Patil P.U**
2. **Kulkarni P.P**
3. **Dani S.P**

Section-II:

1. **Jagdale D.V**
2. **More A.M**

Reviewed By:
Prof. Gangarde A.D

Introduction

1. **About the work book:**
   This workbook is intended to be used by F.Y.B.C.A. (Science) students for the C and RDBMS Assignments in Semester–II. This workbook is designed by considering all the practical concepts / topics mentioned in syllabus.

2. **The objectives of this workbook are:**
   1) Defining the scope of the course.
   2) To bring the uniformity in the practical conduction and implementation in all colleges affiliated to SPPU.
   3) To have continuous assessment of the course and students.
   4) Providing ready reference for the students during practical implementation.
   5) Provide more options to students so that they can have good practice before facing the examination.
   6) Catering to the demand of slow and fast learners and accordingly providing the practice assignments to them.

3. **How to use this workbook:**
   The workbook is divided into two sections. Section-I is related to Advance C assignments and Section-II is related to Introduction to Relational Database Management System.
   The Section-I (Advance C) is divided into Ten assignments. Each Advance C assignment has three SETs. It is mandatory for students to complete the SET A and SET B in given slot.
   The Section-II (Relational database Management System) is divided into fourteen assignments. Each assignment has three SETs. It is mandatory for students to complete SET A and SET B in given slot.

3.1 Instructions to the students

Please read the following instructions carefully and follow them.

1) Students are expected to carry this book every time they come to the lab for computer science practical.
2) Students should prepare oneself beforehand for the Assignment by reading the relevant material.
3) Instructor will specify which problems to solve in the lab during the allotted slot and student should complete them and get verified by the instructor. However student should spend additional hours in Lab and at home to cover as many problems as possible given in this work book.

4) Studentswillbeassessedforeachexerciseonascalefrom0to5.

Not done	0
Incomplete	1
Late Complete	2
Needs improvement	3
Complete	4
Well Done	5

3.2 **Instruction to the Instructors**

1) Explain the assignment and related concepts in around ten minutes using whiteboard if required or by demonstrating the software.
2) You should evaluate each assignment carried out by a student on a scale of 5 as specified above by ticking appropriate box.
3) The value should also be entered on assignment completion page of the respective Lab course.

3.3 **Instructions to the Lab Administrator**

You have to ensure appropriate hardware and software is made available to each student.

The operating system and software requirements on server side and also client side areas given below:
1) Server and Client Side - (Operating System) Linux/Windows
2) Database server – PostgreSQL
3) Turbo C.

Table of Contents for Section-I

Assignment Completion Sheet

Lab Course I			
Advance C Assignments			
Sr. No.	**Assignment Name**	**Marks (out of 5)**	**Teachers Sign**
1	Use of Pre-Processor Directive		
2	Use Of Simple Pointers		
3	Advanced Use Of Pointers		
4	Concept Of Strings, Array Of Strings		
5	String Operations Using Pointers		
6	Command Line Arguments		
7	Structures(Using Array And Functions)		
8	Nested Structures And Unions		
9	Use Of Bitwise Operators		
10	File Handling		
Total (Out of 50)			
Total (Out of 30)			

This is to certify that Mr./Ms.____
Has successfully completed the Advanced programming in C course work for Lab Course-I and has scored ___ Marks out of 30.

Instructor H.O.D. / Coordinator

Internal Examiner External Examiner

Section I – Advanced Programming in C

Assignment No 1:- Study of Preprocessor Directive

1. **File Inclusion Directive (#include)**
2. **Macro**
2. **# error and #pragma directives**
3. **Conditional Compilation**
4. **Predefined macros and Preprocessor operations.**

Practice Programs

1) Example program for #define, #include preprocessors in C language:

```
#include <stdio.h>
#define height 100
#define number 3.14
#define letter 'A'
#define letter_sequence "ABC"
#define special_char '?'
 void main()
{
   printf("value of height     : %d \n", height );
   printf("value of number : %f \n", number );
   printf("value of letter : %c \n", letter );
   printf("value of letter_sequence : %s \n", letter_sequence);
   printf("value of special_char: %c \n", special_char);
 }
```

Output:

```
value of height : 100
value of number : 3.140000
value of letter : A
value of letter_sequence : ABC
value of special_char: ?
```

2) Example program for conditional compilation directives:

a) Example program for #ifdef, #else and #endif in C:

```
#include <stdio.h>
#define RAJU 100
 int main()
{     #ifdef RAJU
        printf("RAJU is defined. So, this line will be added in  this C file\n");
```

```
  #else
  printf("RAJU is not defined\n");
  #endif
  return 0;  }
```

```
Output:   RAJU is defined. So, this line will be added in this C file
```

b) Example program for #ifndef and #endif in C:

```
#include <stdio.h>
#define RAJU 100
int main()
{
  #ifndef SELVA
  {
    printf("SELVA is not defined. So, now we are going to define here\n");
    #define SELVA 300
  }
  #else
    printf("SELVA is already defined in the program");

  #endif
  return 0;       }
```

Output:

```
SELVA is not defined. So, now we are going to define here
```

c) Example program for #if, #else and #endif in C:

```
#include <stdio.h>
#define a 100
int main()
{
  #if (a==100)
       printf("This line will be added in this C file since a = 100\n");
  #else
      printf("This line will be added in this C file since a is not equal to 100\n");
  #endif
  return 0;  }
```

Output:

```
This line will be added in this C file since a = 100
```

3) Example program for #pragma :

a) Illustration to suppress warning

```
    #include<stdio.h>
    #pragma warn –rvl
int main( )
    {
printf("It will not show any warning");
    }
```

Explanation :

- mean suppress warning message .

rvl means : "Function should return a value".

We have specified return type of main as “Integer” (By default it is integer)but we are not going to return a value.
Usually this program will show warning message.
We have suppressed warning already (-rvl) so we won’t get any warning message.
4) Example program for #error :
The #error preprocessor directive indicates error. The compiler gives fatal error if #error directive is found and skips further compilation process.

```
#include<stdio.h>
      #ifndef __MATH_H
      #error First include then compile
      #else
 void main( )
     {    float a;
          a=sqrt(7);
 printf("%f",a);
      }
      #endif
```

1) Predefined Macros

There are some predefined macros which are readily for use in C programming.

Predefined macro	Value
__DATE__	String containing the current date
__FILE__	String containing the file name
__LINE__	Integer representing the current line number
__STDC__	If follows ANSI standard C, then value is a nonzero integer
__TIME__	String containing the current date.

Example #1: predefined Macros

C Program to find the current time

```
#include <stdio.h>
int main()
{
   printf("Current time: %s",   TIME   );    //calculate the current time
}
```

Output: Current time: 19:54:39

2) The following printf statements display the values of the predefined macros **__LINE** , **FILE** , **TIME** , and **DATE__**and print a message indicating the program's conformance to ANSI/ISO standards based on **STDC** :

```
/*     This example illustrates some predefined macros.*/

#pragma langlvl(ANSI)
#include <stdio.h>
#if    STDC__
#   define CONFORM     "conforms"
#else
#   define CONFORM     "does not conform"
#endif
int main(void)
{
  printf("Line %d of file %s has been executed\n", LINE , FILE );
  printf("This file was compiled at %s on %s\n", TIME , DATE );
  printf("This program %s to ANSI/ISO standard C\n", CONFORM);
}
```

3) Stringize (#):
The stringize or number-sign operator ('#'), when used within a macro definition, converts a macro parameter into a string constant. This operator may be used only in a macro that has a specified argument or parameter list.
When the stringize operator immediately precedes the name of one of the macro parameters, the parameter passed to the macro is enclosed within quotation marks and is treated as a string literal.

Example 1:
```
#include <stdio.h>
#define   message_for(a, b)            printf(#a " and " #b ": How are you!\n")
int main(void)
{
   message_for(Carole, Debra);
   return 0;
}
```

Output: Carole and Debra: How are you!

Example 2:
```
#include<stdio.h>
#define string(s) #s
int main(){
    char str[15]= string(World is our );
    printf("%s",str);
    return 0;
}
```

Output: World is our

Explanation : Its intermediate file will look like:

```
int main(){
   char str[15]=”World is our”;
   printf("%s",str);
   return 0;
}
```

Argument of string macro function ‘World is our’ is converted into string by the operator # .Now the string constant “World is our” is replaced the macro call function in line number 4.

4) Token Pasting (##):

The token-pasting operator (##) within a macro definition combines two arguments. It permits two separate tokens in the macro definition to be joined into a single token.
If the name of a macro parameter used in the macro definition is immediately preceded or followed by the token-pasting operator, the macro parameter and the token-pasting operator are replaced by the value of the passed parameter. Text that is adjacent to the token-pasting operator that is not the name of a macro parameter is not affected.

Example1 :

```
#define tokenpaster(n) printf ("token" #n " = %d", token##n)
tokenpaster(34);
```

This example results in the following actual output from the preprocessor:

```
printf ("token34 = %d", token34);
```

This example shows the concatenation of token##n into token34. Both the stringize and the token-pasting operators are used in this example.

Example 2 :

```
#include<stdio.h>
#define merge(p,q,r) p##q##r
int main(){
   int merge(a,b,c)=45;
   printf("%d",abc);
   return 0;
}
```

Output : 45

Explanation:

Arguments a,b,c in merge macro call function is merged in abc by ## operator .So in the intermediate file declaration statement is converted as :

int abc = 45;

5) Multiline macros

By using a the "\" to indicate a line continuation, we can write our macros across multiple lines tomake them a bit more readable.

For instance, we could rewrite swap as

```
#define SWAP(a, b)  {                  \
                    a ^= b;         \
                    b ^= a;         \
                    a ^= b;         \
                }
```

Notice that you do not need a slash at the end of the last line! The slash tells the preprocessor that the macro continues to the next line.
Aside from readability, writing multi-line macros may make it more obvious that you need to use curly braces to surround the body because it's more clear that multiple effects are happening at once.

6) Defined

The special operator defined is used in '#if' and '#elif' expressions to test whether a certain name is defined as a macro. defined *name* and defined (*name*) are both expressions whose value is 1 if *name* is defined as a macro at the current point in the program, and 0 otherwise. Thus, #if defined MACRO is precisely equivalent to #ifdef MACRO.

defined is useful when you wish to test more than one macro for existence at once. For example,

```
#if defined (var) || defined (ns16000)
```

would succeed if either of the names var or ns16000 is defined as a macro.

Conditionals written like this:

```
#if defined BUFSIZE && BUFSIZE >= 1024
```

can generally be simplified to just #if BUFSIZE >= 1024, since if BUFSIZE is not defined, it will be interpreted as having the value zero. If the defined operator appears as a result of a macro expansion, the C standard says the behavior is undefined

Set A

1) Write the Program to implement macros for example:-define constant and array size

2) Write the Program to

a. find maximum of two integers
b. check whether a number is positive ,negative or Zero
d. check given number is even or odd

3) Write the Program to illustrate the use of following using #pragma

Sr.No	Warning Message	Code
1	Code has no effect	eff
2	Function should return a value	rvl
3	Parameter 'parameter' is never used	par
4	Possible use of 'identifier' before definition	def
5	Possibly incorrect assi gnment	pia
6	Unreachable code	rch
7	Non portable pointer conversion	rpt

Set B

1) Write the Program to

a. find minimum of three numbers using nested macros
b. swap two numbers

Set C

1) Write the Program to find cube of a number using nested macros

Assignment Evaluation

0: Not Done [ ]   1: Incomplete [ ]   2: Late Complete [ ]

3: Needs Improvement [ ]   4: Complete [ ]   5: WellDone [ ]

Signature of Teacher

Assignment 2:- Use of Pointer Variables

Pointer :

A pointer provides a way of accessing a variable without referring to the variable directly.

The address of the variable is used.

Syntax:-

```
data_type *var_name;
```

Example : int *p; char *p;

Where, * is used to denote that “p” is pointer variable and not a normal variable

This definition set aside two bytes in which to store the address of an integer variable and gives this storage space the name p.

Key points to remember about pointers in C:

- Normal variable stores the value whereas pointer variable stores the address of the variable.
- The content of the C pointer always be a whole number i.e. address.
- & symbol is used to get the address of the variable.
- * symbol is used to get the value of the variable that the pointer is pointing to.
- If a pointer in C is assigned to NULL, it means it is pointing to nothing.
- The value of null pointer is 0.
- Two pointers can be subtracted to know how many elements are available between these two pointers.
- But, Pointer addition, multiplication, division are not allowed.
- The size of any pointer is 2 byte (for 16 bit compiler).

Valid Pointer Arithmetic Operations

- Adding a number to pointer.
- Subtracting a number form a pointer.
- Incrementing a pointer.
- Decrementing a pointer.
- Subtracting two pointers.
- Comparison on two pointers.

Invalid Pointer Arithmetic Operations

- Addition of two pointers.
- Division of two pointers.
- Multiplication of two pointers.

Example:

```
#include<stdio.h>
#include<conio.h>

void main() {
    int var = 10, *ptr;
    char c_var = 'A', *c_ptr;
  float f_var = 4.65, *f_ptr;

    /* Initialize pointers */
    ptr = &var;
    c_ptr = &c_var;
    f_ptr = &f_var;

    printf("Address of var = %u\n", ptr);
    printf("Address of c_var = %u\n", c_ptr);
    printf("Address of f_var = %u\n\n", f_ptr);

    /* Incrementing pointers */
    ptr++;
    c_ptr++;
    f_ptr++;
    printf("After increment address in ptr = %u\n", ptr);
    printf("After increment address in c_ptr = %u\n", c_ptr);
    printf("After increment address in f_ptr = %u\n\n", f_ptr);

    /* Adding 2 to pointers */
    ptr = ptr + 2;
    c_ptr = c_ptr + 2;
    f_ptr = f_ptr + 2;

    printf("After addition address in ptr = %u\n", ptr);
    printf("After addition address in c_ptr = %u\n", c_ptr);
    printf("After addition address in f_ptr = %u\n\n", f_ptr);

    getch();
    return 0;
}
```

Sample output:

Assume the addresses to be-

```
Address of var = 2293300
Address of c_var = 2293299
Address of f_var = 2293292
```

```
After incrementing address in ptr = 2293304
After incrementing address in c_ptr = 2293300
After incrementing  address in f_ptr = 2293296

After addition address in ptr = 2293312
After addition address in c_ptr = 2293302
After addition address in f_ptr = 2293304
```

Set A

1) Write a program to Interchange values of two numbers using pointers
2) Write a program to display the elements of an array containing n integers in reverse order using pointer
3) Write a program to reverse the elements of an array containing n integers using pointer
4) Write a program to calculate average marks of "n" number of students using pointers and array

.

Set B

1) Write a program to accept an array and a number. Check whether the number is present in the array ,print the index number of the first occurrence of the number.
2) In above program --If number is present calculate the number of occurrences of that number in the array using pointers.
3) Write a program to accept an array sort the given array. (Using pointer)
4) Write a program to accept a matrix of size 3x3 and print the same using pointer.

Set C

1) Write a program to accept a matrix of size 3x3 and print transpose of the matrix using pointer.
2) Write a program to accept two matrices of size 3x3 and print the addition using pointer.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: WellDone [ ]

Signature of Teacher

Assignment 3:- Advanced use of Pointers

C Array of Pointer

Just like array of integers or characters, there can be array of pointers too.
An array of pointers can be declared as :

```
<type> *<name>[<number-of-elements];
```

For example :

```
int *ptr[3];
```

Constant Pointer

A constant pointer is declared as :

```
<type-of-pointer> *const <name-of-pointer>
```

For example :

```
#include<stdio.h>

int main(void)
{
   char ch = 'c';
   char c = 'a';
   char *const ptr = &ch;// A constant pointer
   ptr = &c;// Trying to assign new address to a constant pointer. WRONG!!!!
   return 0;
}
```

Gives a compilation error

Pointer to a constant

Syntax :

```
const <type-of-pointer> *<name-of-pointer>;
```

For example :

```
#include<stdio.h>

int main(void)
{
   char ch = 'c';
const char *ptr = &ch; // A constant pointer 'ptr' pointing to 'ch'
   *ptr = 'a';// WRONG!!! Cannot change the value at address pointed by 'ptr'.
   return 0;
}
```
Compilation error

Multiple Indirection :

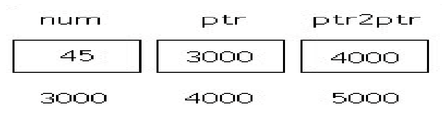
C permits the pointer to point to another pointer. This creates many layers of pointer and therefore called as multiple indirection. A pointer to a pointer has declaration similar to that of a normal pointer but have more asterisk(*) before them. This indicates the depth of the pointer

For Example: Double Pointer

Declaration :

```
int  **ptr2ptr;
```
Example:
```
int num = 45,*ptr,**ptr2ptr;
Ptr=&num;
Ptr2ptr=&ptr;
```

Diagrammatic Representation:



Statement	What will be the Output ?
*ptr	45
**ptr2ptr	45
ptr	&num
ptr2ptr	&ptr

Passing pointer to function and Returning pointer from function

A function can also return a pointer to the calling function. In this case you must be careful, because local variables of function doesn't live outside the function, hence if you return a pointer connected to a local variable, that pointer be will pointing to nothing when function ends.

Example :-
```
#include <stdio.h>
#include <conio.h>
int* larger(int*, int*);
void main()
{
 int a=15;
 int b=92;
 int *p;
 p=larger(&a, &b);
 printf("%d is larger",*p);
}
int* larger(int *x, int *y)
{
 if(*x > *y)
  return x;
 else
  return y;}
```

Pointer to functions

It is possible to declare a pointer pointing to a function which can then be used as an argument in another function. A pointer to a function is declared as follows,

type (***pointer-name**)(*parameter*);

Example :

```
int (*sum)();   //legal declaraction of pointer to function
int *sum();   //This is not a declaraction of pointer to function
```

Example:-

```
#include <stdio.h>
#include <conio.h>

int sum(int x, int y)
{
 return x+y;
}

int main( )
{
 int (*fp)(int, int);
 fp = sum;
 int s = fp(10, 15);
 printf("Sum is %d",s);
 getch();
 return 0;
}
```

Output : 25

Function pointer

void *(*foo) (int*) ;

It appears complex but it is very simple. In this case **(*foo)** is a pointer to the function, whose return type is **void*** and argument is of **int*** type.

Example Program:

```
/*program  to add two numbers using function pointer */
#include<stdio.h>

int sum (int n1,int n2);
int main()
{
int num1=10;
int  num2=20;
int result;

int*(*ptr)(int,int);
ptr=&sum;
result=(*ptr)(num1,num2);
printf(Addition is : %d",result);
return 0;
}

int sum(int n1,int n2)
{
    return(n1+n2)
}
```

Output :Addition:20

Dynamic Memory Allocation:

The functions used are:

malloc()- Allocates requested size of bytes and returns a pointer pointing to first byte of allocated space
Syntax:-
ptr= (cast type *)malloc(Element_size);

calloc()-Allocates multiple blocks of memory each of same size of bytes and returns a pointer pointing to first byte of allocated space .Sets all bytes to zero.
Syntax:-
ptr= (cast type *)calloc(n,Element_size);

realloc()- reallocate previously allocated memory
syntax:-
ptr= (cast type *)realloc(pointer,new_Element_size);

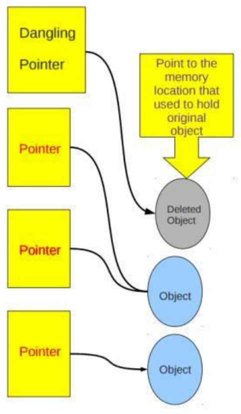
free()- to free the previously allocated memory
syntax:-
free(pointer);

```
/*Program to create memory for int , char and float variables at run time*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
        int *ivar;
       char *cvar;
       float *fvar;

// Allocating memory dynamically
ivar=(int *)malloc(1*sizeof(int));
cvar=(char *)malloc(1*sizeof(char));
fvar=(float *)malloc(1*sizeof(float));

printf("Enter the integer  value :");
scanf("%d",&ivar);

printf("Enter the character value :");
scanf("%c",&cvar);

printf("Enter the float value :");
scanf("%f",&fvar);

printf("The  inputted value are   %d  %c  %f :",ivar,cvar,fvar);


/* free the allocated memory*/
free(ivar);
free(cvar);
free(fvar);

getch();
}
```

What is Dangling Pointer?

While programming, we use pointers that contain memory addresses of data objects. Dangling pointer is a pointer that points to the memory location even after its de-allocation. Or we can say that it is pointer that does not point to a valid data object of the appropriate type. The memory location pointed by dangling pointer is known as dangling reference.

Now if we access the data stored at that memory location using the dangling pointer then it will result in program crash or an unpredictable behavior.



Let's take an example to understand this.

Cause of Dangling Pointer in C

```
void function(){
   int *ptr = (int *)malloc(SIZE);
   . . . . . .
   . . . . . .
   free(ptr);    //ptr now becomes dangling pointer which is pointing to dangling reference
}
```

In above example we first allocated a memory and stored its address in ptr. After executing few statements we de-allocated the memory. Now still ptr is pointing to same memory address so it becomes dangling pointer.

How to Solve Dangling Pointer Problem in C?

To solve this problem just assign NULL to the pointer after the de-allocation of memory that it was pointing. It means now pointer is not pointing to any memory address.

```
void function(){
   int *ptr = (int *)malloc(SIZE);
   . . . . . .
   . . . . . .
   free(ptr);   //ptr now becomes dangling pointer which is pointing to dangling
reference
   ptr=NULL;    //now ptr is not dangling pointer
}
```

What is Memory leak in C and how can we avoid it.

Memory leak happens when programmer allocated memory in heap but don't release it back to the heap. Memory leak reduces the available memory for program and as a result the performance of program reduces.
Here is an example of *memory leak*:

```
#include <stdio .h>
#include <stdlib.h>
 Void getSometing(){
   /* Dynamically declare memory for an integer array of 10 elements */
   int*array = (int*) malloc(10*sizeof(int));
   /* Do something and return without releasing allocated memory */
   return;
}

int main() {
   int i;
   for(i = 0; i<100; i++){
      getSometing();
   }
```

```
    return0;
}
```

In above program, function getSometing( ) dynamically allocates memory an array but then returns without de-allocating it. Every time getSometing( ) function is called it reduces the available memory for the program. To avoid memory leaks, getSometing( ) function should release allocated memory using free.

```
voidgetSometing(){
   /* Dynamically declare memory for an integer array of 10 elements */
   int*array = (int*) malloc(10*sizeof(int));
   /* Do something and release allocated memory */
   free(array);
   return;
}
```

SET A

1) Write a program to multiply two numbers using function pointer
2) Write a Program to accept an array and print the same using double pointer
3) Write a program to calculate average of array of n numbers .Pass the array to a function and use pointers
4) Write a program to accept an array and a number .Write a function to find the number of occurrences of number in the array.(using pointer)

SET B

1) Write a program to read 1 D array of "n" elements and print the inputted array element (using dynamic memory allocation)
2) Write a program to find sum of n elements entered by user. To perform this, allocate memory dynamically using malloc() function
3) Write a program to find sum and average of n elements entered by user. To perform this, allocate memory dynamically using calloc() function.
4) Write a program to find largest among "n" numbers using dynamic memory allocation.

SET C

1) Write a program to accept a 2D array and print the addition of all elements (allocate memory at run time)

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Teacher

Assignment 4:- Concept of Strings , Array of Strings

String is a sequence of characters that is treated as a single data item and terminated by null character '\0'. Remember that C language does not support strings as a data type. A **string** is actually one-dimensional array of characters in C language. These are often used to create meaningful and readable programs.

Declaring and Initializing a string variables

```
char name[13] = "C Prgramming";       // valid character array initialization

char name[10] = {'L','e','s','s','o','n','s','\0'};     // valid initialization
```

C supports a format specification known as the **edit set conversion code %[..]** that can be used to read a line containing a variety of characters, including white spaces.

```
#include<stdio.h>
#include<string.h>
void main()
{   char str[20];
    printf("Enter a string");
    scanf("%[^\n]", &str);  //scanning the whole string, including the white spaces
    printf("%s", str);
}
```

Another method to read character string with white spaces from terminal is by using the gets() function.

```
char text[20];
gets(text);
printf("%s", text);
```

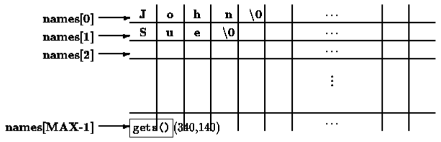
Arrays of String

Besides data base applications, another common application of two dimensional arrays is to store an array of strings. In this section we see how an array of strings can be declared and operations such as reading, printing and sorting can be performed on them.

A string is an array of characters; so, an array of strings is an array of arrays of characters. Of course, the maximum size is the same for all the strings stored in a two dimensional array. We can declare a two dimensional character array of MAX strings of size SIZE as follows:

```
char names[MAX][SIZE];
```

Since names is an array of character arrays, names[i] is the character array, i.e. it points to the character array or string, and may be used as a string of maximum size SIZE - 1. As usual with strings, a NULL character must terminate each character string in the array. We can think of an array of strings as a table of strings, where each row of the table is a string as shown in Figure below



Set A

1) Write a program to find the length of a string.
2) Write a program to copy a string into another.
3) Write a program to reverse a string by passing it to a function.
4) Write a program to check whether a given string is palindrome or not. (Write a function which accepts a string and returns 1 if it is palindrome and 0 otherwise)
5) Write a program to concatenate two strings.

Set B

1) Write a program to find the number of vowels ,consonants, digits and white space in a string.
2) Write a program that accepts names of n cities and write functions for the following:
   a) Search for a city
   b) Display the longest names
3) Write a program which accepts a sentence from the user and replaces all lower case letters by uppercase letters.
4) Write a program to find the First Capital Letter in a String .write a function iscap() to find the first capital letter.
5) Write a program to remove all other characters in a string except alphabets.

Set C

1) Write a program to accept a word and a string .Remove / delete the given word from a string.

Example:- if word is= “Hello ” and the String is “Hello All Well Come”

The output is:- “All Well Come”

2) Write a program to print the words ending with letter d in the given sentence(multiword string).

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: WellDone [ ]

Signature of Teacher

Assignment No 5:- String operations using Pointers

To demonstrate string operations using pointers

C – string.h library functions

All C inbuilt functions which are declared in string.h header file are given below. The source code for string.h header file is also given below for your reference.

List of inbuilt C functions in string.h file:

String functions	Description
strcat ( )	Concatenates str2 at the end of str1
strncat ( )	Appends a portion of string to another
strcpy ( )	Copies str2 into str1
strncpy ( )	Copies given number of characters of one string to another
strlen ( )	Gives the length of str1
strcmp ( )	Returns 0 if str1 is same as str2. Returns <0 if strl < str2. Returns >0 if str1 > str2
strcmpi ( )	Same as strcmp() function. But, this function negotiates case. "A" and "a" are treated as same.
strchr ( )	Returns pointer to first occurrence of char in str1
strrchr ( )	last occurrence of given character in a string is found
strstr ( )	Returns pointer to first occurrence of str2 in str1
strrstr ( )	Returns pointer to last occurrence of str2 in str1
strdup ( )	Duplicates the string
strlwr ( )	Converts string to lowercase
strupr ( )	Converts string to uppercase
strrev ( )	Reverses the given string
strset ( )	Sets all character in a string to given character
strnset ( )	It sets the portion of characters in a string to given character
strtok ( )	Tokenizing given string using delimiter

Example program for strcat( ) function in C:

In this program, two strings "fresh2refresh" and "C tutorial" are concatenated using strcat( ) function and result is displayed as "C tutorial fresh2refresh".

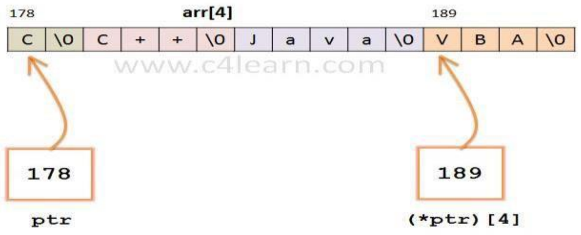
```
#include<stdio.h>
#include<string.h>
int main( )
{
	char source[]="For Student";
```

```
        char target[]="C Tutorial";

        printf("\n Source String =%s",source);
        printf("\n Target String =%s",target);
        strcat(target,source);
         printf("\n Target SAtring after strcat()=%s",target);
}
```

```
Sample Output:
Source String                          =For Student
Target String                   = C Tutorial
Target String after strcat( )=C Tutorial For Student
```

Pointer to array of strings

A pointer which is pointing to an array whose content are strings,is known as pointer to array of strings.



With reference to above example

1. ptr : It is pointer to array of string of size 4.
2. array[4] : It is an array and its content are string.

/*program to print the Address of the Character Array */

```
#include<stdio.h>
int main()
{
int i;
char * arr[4]={"C","C++","JAVA","VBA"};
char *(*ptr)[4]=&arr[0];
for(i=0;i<4;i++)
    printf("Address of String %d : %u\n",i+1,(*ptr)[i]);
retutn 0;
```

```
Output:
Address of string1=178
Address of String2=180
Address of String3=184
Address of String4=189
```

Set A

1) Write a program to Calculate Length of the String using Pointer.
2) Write a program to print Contents of 2-D character array(using pointer to array of string)Refer to above example.
3) Write a function similar to strlen that can handle unterminated strings.
Hint: you will need to know and pass in the length of the string.
4) Write a program to compare only left most ‘n’ characters from the given string .Accept n and string to be compared from the user.(using pointer)
5) Write a program to
    -compare two strings using library function
    -sets the portion of characters in a string to given character using library function

Set B

1) Write a program to compare two strings. If they are not equal display their length and if equal concatenate them
2) Write a program to pass two strings to user defined function and copy one string to another using pointer
3) Write a program to reverse string, without using another string variable.
4) Write a program to accept a string and a substring and check if the substring is present in the given string

Set C

1) Write a program to Count number of words in the given sentence .
2) Write a program to concatenate only the leftmost n characters from source with the destination string.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: WellDone [ ]

Signature of Teacher

Assignment No 6:-Command Line Arguments

Command line arguments in C:

main() function of a C program accepts arguments from command line or from other shell scripts by following commands. They are,

- argc
- argv[]

where,

argc – Number of arguments in the command line including program name
argv[] – This is carrying all the arguments

Example

```
#include<stdio.h>
#include<conio.h>
        void main(int argc,char* argv[])
        { clrscr();
          printf("\n Program name  : %s \n", argv[0]);
          printf("1st arg  : %s \n", argv[1]);
          printf("2nd arg  : %s \n", argv[2]);
          printf("3rd arg  : %s \n", argv[3]);
          printf("4th arg  : %s \n", argv[4]);
          printf("5th arg  : %s \n", argv[5]);
        getch();}
```

Set A

1) Write a program to display the arguments passed using command line argument(refer to above example).
2) Write a program to add two numbers using Command Line Arguments.

Set B

1) Write a program to calculate the factorial of one number by using the command line argument.
2) Write a program to Generate Fibonacci Series of N Numbers using Command-Line Argument.

Set C

1)Write a program to accept three integers as command line arguments and find the minimum, maximum and average of the three numbers. Display error message if the number of arguments entered are invalid.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: WellDone [ ]

Signature of Teacher

Assignment No 7:-Structures (Using Arrays and Functions)
Structure

C Structure is a collection of different data types which are grouped together and each element in a C structure is called member.

- If you want to access structure members in C, structure variable should be declared.
- Many structure variables can be declared for same structure and memory will be allocated for each separately.
- It is a best practice to initialize a structure to null while declaring, if we don't assign any values to structure members.

Array of Structures

- C Structure is collection of different datatypes (variables) which are grouped together. Whereas, array of structures is nothing but collection of structures. This is also called as structure array in C.

Passing structure to a function

- A structure can be passed to any function from main function or from any sub function.
- Structure definition will be available within the function only.
- It won't be available to other functions unless it is passed to those functions by value or by address(reference).
- Else, we have to declare structure variable as global variable. That means, structure variable should be declared outside the main function. So, this structure will be visible to all the functions in a C program.

Passing structure to function in C:

It can be done in 3 ways.

1. Passing structure to a function by value
2. Passing structure to a function by address(reference)
3. No need to pass a structure – Declare structure variable as global

Structure using Pointer

C structure can be accessed in 2 ways in a C program. They are,

1. Using normal structure variable
2. Using pointer variable

Dot(.) operator is used to access the data using normal structure variable and arrow (->) is used to access the data using pointer variable.

Set A

1) Write a program to store and access "id, name and percentage" for one student.

2) Write a program to create student structure having fields roll_no, stud_name, mark1, mark2, mark3.
Calculate the total and average of marks

3) Write a program to create student employee having field emp_id, emp_name, designation.
Pass this entire structure to function and display the structure elements.

4) Write a program to copy one structure to another structure .(using pointer)

Set B

1) Write a program to store and access "id, name and percentage" for 3 students.(array of structures)

2) Write a program to create a student structure having fields roll_no, stud_name and address.
Accept the details of 'n' students into the structure, rearrange the data in alphabetical order of
stud_name and display the result.

3)Create an employee structure(eno, ename, salary). Write a menu driven program to perform the following operations (using function)

a. Add employee
b. Display all employees having salary>10000

Set C

1)Write a menu driven program in 'C' that shows the working of a library.
The menu option should be

- Add book information.
- Display book information.
- List all books of given author.
- List the count of books in the library.
- Exit.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Teacher

Assignment No 8 :-Nested Structures and Unions

Nested Structure

- Nested structure in C is nothing but structure within structure. One structure can be declared inside other structure as we declare structure members inside a structure.
- The structure variables can be a normal structure variable or a pointer variable to access the data. You can learn below concepts in this section.

1. Structure within structure in C using normal variable
2. Structure within structure in C using pointer variable

Union

Union is also like structure, i.e. collection of different data types which are grouped together. Each element in a union is called member.

- Union and structure in C are same in concepts, except allocating memory for their members.
- Structure allocates storage space for all its members separately.
- Whereas, Union allocates one common storage space for all its members
- We can access only one member of union at a time. We can't access all member values at the same time in union. But, structure can access all member values at the same time. This is because,Union allocates one common storage space for all its members. Where as Structure allocates storage space for all its members separately.
- Many union variables can be created in a program and memory will be allocated for each union variable separately.

pointer to union

pointer is special kind of variable which is capable of storing the address of a variable in c programming. Pointer which stores address of union is called as pointer to union.

Set A

1)Write a program to declare a structure "employee"(name,age,salary) which contains another structure "address"(house number, street) as member variable .Accept the details of one employee and display it.(using normal variable)

2) Write a program to declare a structure "employee"(name,age,salary) which contains another structure "address"(house number,street) as member variable.Accept the details of one employee and display it.(using pointer variable)

3) Write a program to to store and access " name, subject and percentage" for two student.(using union)

4) Write a program to store and access “ name, subject and percentage” for one student.(using pointer to union)

Set B

1) Write a program to create a union for a library book with the following details (id , title , publisher , cost).If the code is 1 store the number of copies,if code= 2 store the issue month name and if code=3 store the edition number.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Teacher

Assignment No 9 :-Use of Bitwise Operators

```
* C Program to demonstrate use of bitwise operators */
#include<stdio.h>
int main()
{
    unsigned char a = 5, b = 9; // a = 5(00000101), b = 9(00001001)
    printf("a = %d, b = %d\n", a, b);
    printf("a&b = %d\n", a&b); // The result is 00000001
    printf("a|b = %d\n", a|b);  // The result is 00001101
    printf("a^b = %d\n", a^b); // The result is 00001100
    printf("~a = %d\n", a = ~a);    // The result is 11111010
    printf("b<<1 = %d\n", b<<1);  // The result is 00010010
    printf("b>>1 = %d\n", b>>1);  // The result is 00000100
    return 0;
}
```

Output:

```
a = 5, b = 9
a&b = 1
a|b = 13
a^b = 12
~a = 250
b1 = 4
```

Following are interesting facts about bitwise operators.

1) The left shift and right shift operators should not be used for negative numbers
The result of is undefined behabiour if any of the operands is a negative number. For example results of both -1 << 1 and 1 << -1 is undefined. Also, if the number is shifted more than the size of integer, the behaviour is undefined. For example, 1 << 33 is undefined if integers are stored using 32 bits

The bitwise XOR operator is the most useful operator from technical interview perspective. It is used in many problems. A simple example could be “Given a set of numbers where all elements occur even number of times except one number, find the odd occurring number” This problem can be efficiently solved by just doing XOR of all numbers.

```
// Function to return the only odd occurring element
int findOdd(int arr[], int n) {
   int res = 0, i;
   for (i = 0; i < n; i++)
     res ^= arr[i];
   return res;
}
```

```
int main(void) {
   int arr[] = {12, 12, 14, 90, 14, 14, 14};
   int n = sizeof(arr)/sizeof(arr[0]);
   printf ("The odd occurring element is %d ", findOdd(arr, n));
   return 0;
}
// Output: The odd occurring element is 90
```

<u>Set A</u>

1) Write a program to swap two numbers using bitwise operators
2) write a menu driven program in 'C' to accept a number from user and perform following operations on it
    i) Right Shift
    ii) Left Shift
    iii) One's complement

<u>Set B</u>

1) Given a list of n-1 integers and these integers are in the range of 1 to n. There are no duplicates in list. One of the integers is missing in the list. Write a program to find the missing integer.

Example:

I/P [1, 2, 4, ,6, 3, 7, 8]
O/P 5

2) Write a program to check whether the given Number is Palindrome or not using Bitwise Operator

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Teacher

Assignment No 10 :-File Handling

File Handling(Text Files)

You should read the following topics before starting this exercise

1. Concept of streams
2. Declaring a file pointer
3. Opening and closing files
4. File opening modes
5. Random access to files

Operations performed	**Syntax**	**Example**
Declaring File pointer	FILE * pointer;	FILE *fp;
Opening a File	fopen("filename",mode); where mode = "r", "w", "a", "r+", "w+", "a+"	fp=fopen("a.txt", "r");

Checking for successful open	if (pointer==NULL)	if(fp==NULL)
		exit(0);

Checking for end of file
 	
Closing a File
	feof
	
	fclose(pointer);
	fcloseall();
		if(feof(fp))
		printf("File has ended");
		fclose(fp);

Character I/O	fgetc, fscanf
	fputc, fprintf
		ch=fgetc(fp);
		fscanf(fp, "%c",&ch);
		fputc(fp,ch);

String I/O	fgets, fscanf
	fputs, fprintf
		fgets(fp,str,80);
		fscanf(fp, "%s",str);

Reading and writing formatted data	fscanf
	fprintf
		fscanf(fp, "%d%s",&num,str);
		fprintf(fp, "%d\t%s\n", num, str);

Random access to files	ftell, fseek, rewind	fseek(fp,0,SEEK_END); /* end of file*/ long int size = ftell(fp);

File Handling (Binary File)

In binary files, information is written in the form of binary . All data is written and read with no interpretation and separation i.e. there are no special characters to mark end of line and end of file.

I/O operations on binary files

Reading from a binary file	fread(address,size-of-element,number of elements,pointer);	fread (&num,sizeof(int),1,fp);
		fread (&emp,sizeof(emp),1,fp);
		fread(arr,sizeof(int),10,fp) ;

Writing to a binary file	fwrite(address,size-of-element,number of elements,pointer);	fwrite (&num,sizeof(int),1,fp);
		fwrite (&emp,sizeof(emp),1,fp);

Set A

1) Write a program to create a file, read its contents and display on screen with each case of character reversed.

2) Write a program to create a file called emp.rec and store information about a person in terms of his name, age and salary.

3) Write a program to accept two filenames as command line arguments. Copy the contents of the first file to the second such that the case of all alphabets is reversed.

4) Write a program to write data of 5 employees to a binary file and then read the file.

5) Write a program to count the number of lines in a file.

Set B

1) Write a program to delete specific line from a file

2) Write a program to Compare two Binary Files, Printing the First Byte Position where they Differ

3) Write a menu driven program to create a structure student (roll number, name, percentage) . Perform the following operations on a binary file- “student.dat”.

- Add a student (Note: Students should be assigned roll numbers consecutively)
- Search Student
  - according to roll number
- Display all students

Set C

1)Write a menu driven program for a simple text editor to perform the following operations on a file, which contains lines of text.

i. Display the file
ii. Copy m lines from position n to p
iii. Delete m lines from position p
iv. Modify the nth line
v. Add n lines

2) Create two binary files such that they contain roll numbers, names and percentages. The percentages are in ascending orders. Merge these two into the third file such that the third file still remains sorted on percentage. Accept the three filenames as command line arguments

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Teacher

Savitribai Phule Pune University

F. Y. B. C. A. (Science) Semester-II

Lab Course – II

Workbook

Name: ____

College Name:____

Roll No.:_________________Division:____________________

Academic Year:________________

Savitribai Phule Pune University

Section-II

F. Y. B. C. A. (Science)

SEMESTER II

BCA - 206

Lab Course - II
Relational Database Management System Assignments

Table of Contents

Assignment Completion Sheet

Lab Course II			
Advanced Relational Database Management System Assignments			
Sr. No.	**Assignment Name**	**Marks (out of 5)**	**Instructor Sign**
1	Case study – ER diagram		
2	Case study – ER diagram (with generalization)		
3	Case study – ER diagram (with aggregation)		
4	Using PostgreSQL (demo of PostgrSQL)		
5	Data Definition queries (Create)		
6	Data Definition queries (Alter)		
7	Data Definition queries (Drop)		
8	Simple queries (Select)		
9	Queries with join		
10	Aggregate queries (Group by and Having)		
11	Nested Queries		
12	Data Manipulation queries (Insert)		
13	Data Manipulation queries (Delete)		
14	Data Manipulation queries (Update)		
Total (Out of 70)			
Total (Out of 30)			

This is to certify that Mr./Ms.____
Has successfully completed the RDBMS course work for Lab Course II and has scored______Marks out of 30.

Instructor **H.O. D / Coordinator**

Internal Examiner **External Examiner**

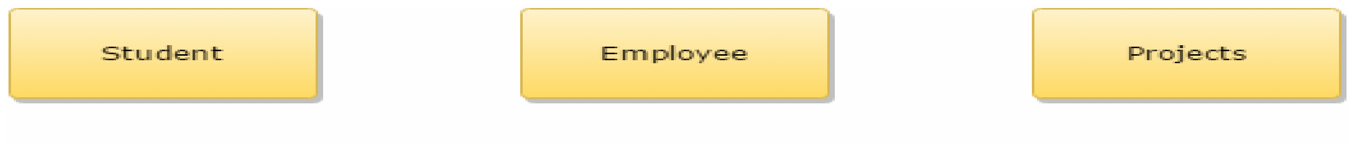
Assignment No. 1

What is Entity Relationship Diagram (ER-Diagram)?

ER-Diagram is a pictorial representation of data that describes how data is communicated and related to each other. Any object, such as entities, attributes of an entity, sets of relationship and other attributes of relationship can be characterized with the help of the ER diagram.

Entities:

They are represented using the rectangle shape box. These rectangles are named with the entity set they represent.



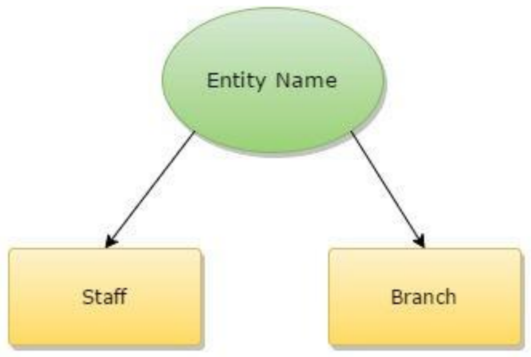
Entities

ER modeling is a top-down structure to database design that begins with identifying the important data called entities and relationships in combination between the data that must be characterized in the model. Then database model designers can add more details such as the information they want to hold about the entities and relationships which are the attributes and any constraints on the entities, relationships, and attributes. ER modeling is an important technique for any database designer to master and forms the basis of the methodology.

- **Entity type:** It is a group of objects with the same properties that are identified by the enterprise as having an independent existence. The basic concept of the ER model is the entity type that is used to represent a group of 'objects' in the 'real world' with the same properties. An entity type has an independent existence within a database.
- **Entity occurrence:** A uniquely identifiable object of an entity type.

Diagrammatic Representation of Entity Types

Each entity type is shown as a rectangle labeled with the name of the entity, which is normally a singular noun.

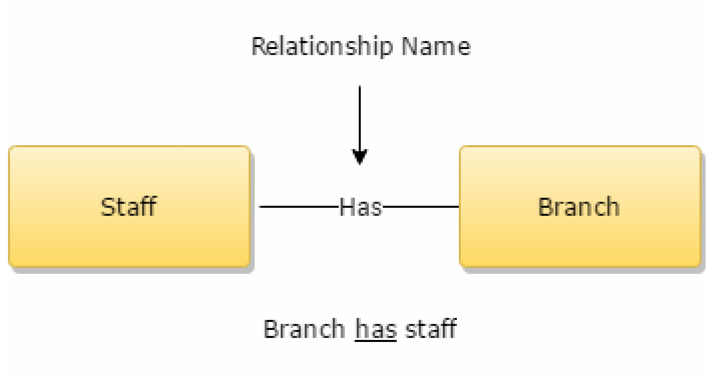


Diagrammatic representation of the Staff and Branch entity types.

What is Relationship Type?

A relationship type is a set of associations between one or more participating entity types. Each relationship type is given a name that describes its function.

Here is a diagram showing how relationships are formed in a database.



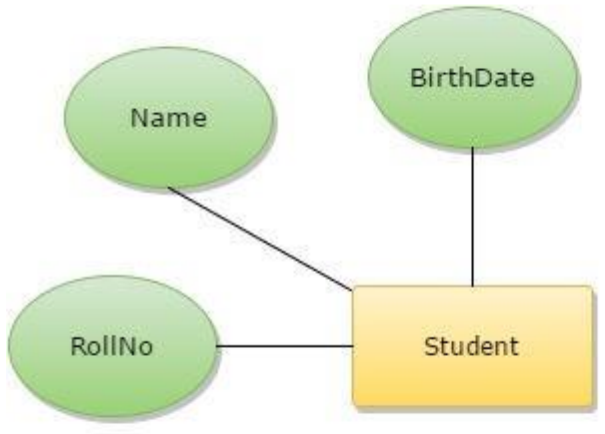
What is degree of Relationship?

The entities occupied in a particular relationship type are referred to as participants in that relationship. The number of participants involved in a relationship type is termed as the degree of that relationship.

In the above figured example “Branch has staff”, there is a relationship between two participating entities. A relationship of degree two is called binary degree (relationship).

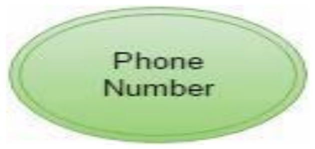
What are Attributes?

Attributes are the properties of entities that are represented by means of ellipse shaped figures. Every elliptical figure represents one attribute and is directly connected to its entity (which is represented as rectangle).



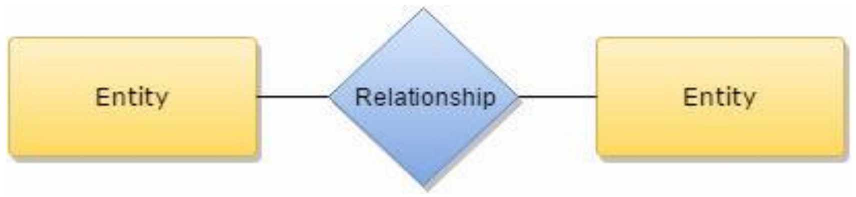
Attributes

It is to be noted that multi-valued attributes are represented using double ellipse like this:



Relationships

Relationships are represented by diamond-shaped box. All the entities (rectangle shaped) participating in a relationship gets connected using a line.



There are four types of relationships. These are:

- **One-to-one:** When only a single instance of an entity is associated with the relationship, it is termed as ‘1:1’.
- **One-to-many:** When more than one instance of an entity is related and linked with a relationship, it is termed as ‘1:N’.
- **Many-to-one:** When more than one instance of entity is linked with the relationship, it is termed as ‘N:1’.
- **Many-to-many:** When more than one instance of an entity on the left and more than one instance of an entity on the right can be linked with the relationship, then it is termed as N:N relationship.

Set A

Suppose you are given the following requirements for a simple database for the National Hockey League (NHL):

- the NHL has many teams,
- each team has a name, a city, a coach, a captain, and a set of players,
- each player belongs to only one team,
- each player has a name, a position (such as left wing or goalie), a skill level, and a set of injury records,
- a team captain is also a player,
- a game is played between two teams (referred to as host_team and guest_team) and has a date (such as May 11th, 1999) and a score (such as 4 to 2).

Consider the case study given above and find out entities and their attributes.

Set B

Find different set of entities and their attributes for online bookstore

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: WellDone [ ]

Signature of Teacher

Assignment no 2

The ER model supported with additional semantic concepts is called the Enhanced Entity-Relationship (EER) model. There are three of the most important and useful added concepts of the EER model, namely specialization/generalization, aggregation, and composition. In this chapter you will learn about the main two important concepts. These are:

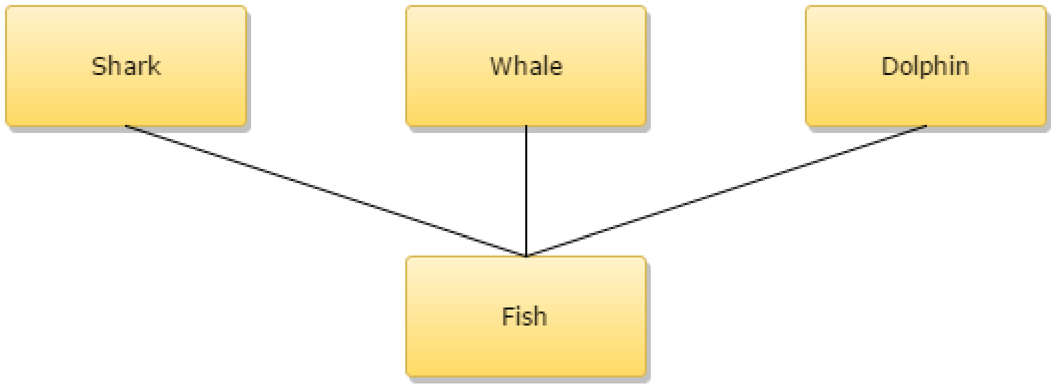
- Generalization Aggregation

What are Generalization / Specialization?

The concept of generalization (specialization) is associated with special types of entities known as super classes and subclasses, and the process of attribute inheritance. Database managers begin this section by defining what super classes and subclasses are and by examining super class/subclass relationships. The ER Model has the capability of articulating database entities in a conceptual hierarchical manner. As the hierarchy goes up, it generalizes the view of entities and as you go deep in the hierarchy, it will provide with the detail of every entity included. Going up in this structure is called generalization, where entities are associated together to represent a more generalized view. Generalization is a bottom-up approach.

In generalization, a number of entities are accommodated together into one generalized entity or category based on their similar characteristics. In the below mentioned figure, whale, shark and dolphin are generalized as fish, i.e. they have been categorized as the fish.

- **Super-class**: An entity type that includes one or more dissimilar sub-groupings of its occurrences that is required to be represented in a data model.
- **Sub-class**: A distinct sub-grouping of occurrences of an entity type that require to be represented in a data model.



Generalization

Super-class/Subclass Relationships

Each member of a sub class is also a member of the super class i.e. the entity in the sub class is the same entity in the super class, but has a different role. The relationship between a super class and a sub class is one-to-one (1:1) and is termed as a super-class/sub-class relationship.

Set A

Refer to the case studies given in assignment no 1

1. Find the relationships among all entities from set A
2. Find the relationships among all entities from set B

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: WellDone [ ]

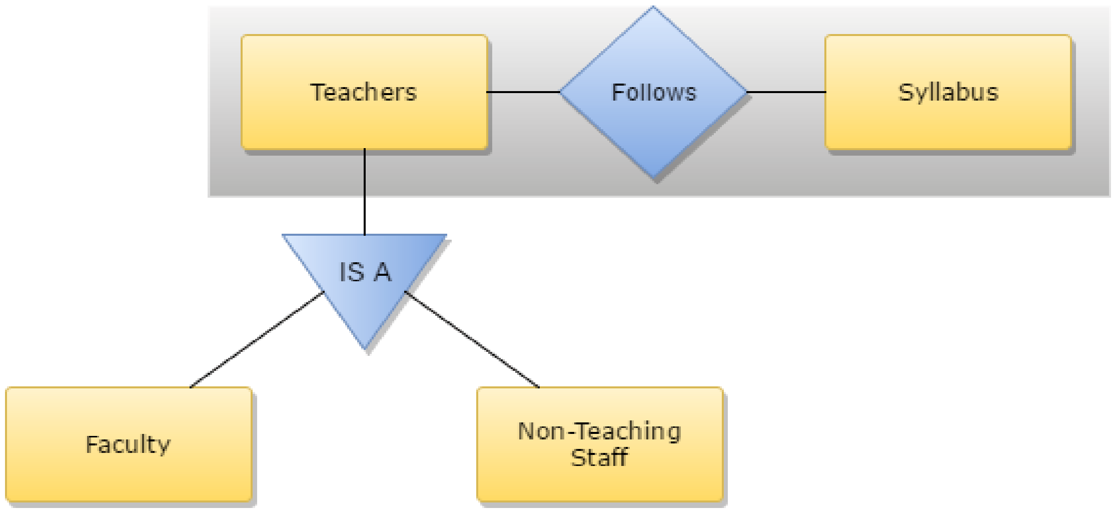
Signature of Teacher

Assignment no.3

What is Aggregation?

A relationship represents a connection between two entity types that are conceptually at the same level. Sometimes you may want to model a 'has-a', 'is-a' or 'is-part-of' relationship, in which one entity represents a larger entity (the 'whole') that will consist of smaller entities (the 'parts'). This special kind of relationship is termed as an aggregation. Aggregation does not change the meaning of navigation and routing across the relationship between the whole and its parts.

An example of an aggregation is the 'Teacher' entity following the 'syllabus' entity act as a single entity in the relationship. In simple words, aggregation is a process where the relation between two entities is treated as a single entity.



Specialization and Aggregation

Set A

Draw an Entity-Relationship diagram for the National Hockey League (NHL):

- the NHL has many teams,
- each team has a name, a city, a coach, a captain, and a set of players,
- each player belongs to only one team,
- each player has a name, a position (such as left wing or goalie), a skill level, and a set of injury records,
- a team captain is also a player,
- a game is played between two teams (referred to as host_team and guest_team) and has a date (such as May 11th, 1999) and a score (such as 4 to 2).

Set B

Consider a database used to record the marks that students get in different exams of different course offerings.
a) Construct an E-R diagram that models exams as entities, and uses a ternary relationship, for the above database.

b) Construct an alternative E-R diagram that uses only a binary relationship between students and course-offerings. Make sure that only one relationship exists between a particular student and course-offering pair, yet you can represent the marks that a student gets in different exams of a course offering.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: WellDone [ ]

Signature of Teacher

Assignment no.4

Using Postgresql (Demo of Postgresql)

Installalation of PostgreSQL on Windows

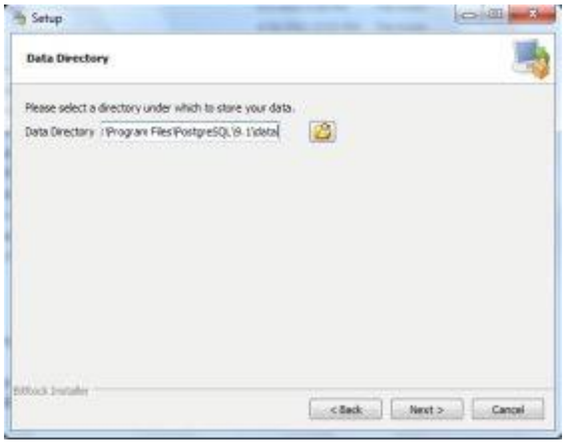
Download PostgreSQL

To download PostgreSQL to install it on Windows 7, please visit the following web page : http://www.postgresql.org/download/windows and click on the "Download" link under "One click installer". The downloaded package will install PostgreSQL Server and pgadmin III GUI to manage PostgreSQL Server and StackBuilder which can be used to download drivers and tools for PostgreSQL Server.

Once you click on the said "Download" link, it will take you to another page from where you need to select the package depending upon your OS platform. So, for installing PostgreSQL on 32 bit Windows 7, select "Win x86-32". If you are using a 64 bit OS, select "Win x86-64". That will start the download process and depending up on your connection speed, take a while to get downloaded.

Make sure you have turned Third Party AntiVirus off while installing.

Once the download is finished, run the postgresql-9.1.1-1-windows.exe file and select the location where you want to install it. By default, it is installed within Program Files folder. Then it asks you to enter a password. Keep the port as default. When asked for "Locale", we have selected "English, United States". It will take a while to install PostgreSQL on your system.

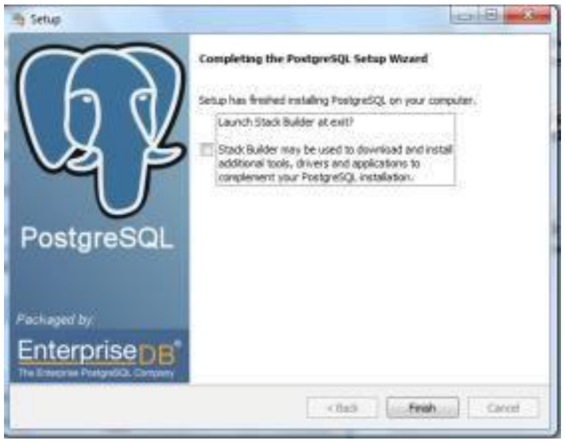


Data Directory

Please select a directory under which to store your data.

Data Directory :\Program Files\PostgreSQL\9.1\data

On completion of the installation process, you will get the following screen.



Completing the PostgreSQL Setup Wizard

Setup has finished installing PostgreSQL on your computer.

Launch Stack Builder at exit?

☐ Stack Builder may be used to download and install additional tools, drivers and applications to complement your PostgreSQL installation.

After the installation process is completed, you can access pgAdmin III, psql, StackBuilder and PostgreSQL shell from your Program Menu under PostgreSQL 9.1.

Connect to PostgreSQL from the command line

Running the PostgreSQL interactive terminal program, called psql, which allows you to interactively enter, edit, and execute SQL commands. At the time of installing postgres to your operating system, it creates an "initial DB" and starts the postgres server domain running. Typically initdb creates a table named "postgres" owned by user "current logged in user name"

At the command line in your operating system, type the following command.

Debian based systems like Ubuntu :
Connect/login as root -

```
user@user-pc:~$ sudo -i -u postgres
postgres@user-pc:~$ psql
psql (9.3.5, server 9.3.6)
Type "help" for help.
```

Redhat based systems like Centos / Fedora :
Connect/login as root -

```
user@user-pc:~$ su - postgres
user@user-pc:~$ psql
psql (9.3.6)
Type "help" for help.
```

Windows :
In windows, current user doesn't matter

```
C:\Program Files\PostgreSQL\9.4\bin>psql -U postgres
Password for user postgres:
psql (9.4.1)
Type "help" for help.

postgres=#
```

After accessing a PostgreSQL database, you can run SQL queries and more. Here are some common *psql* commands

- To view help for *psql* commands, type \?.
- To view help for SQL commands, type \h.
- To view information about the current database connection, type \conninfo.
- To list the database's tables and their respective owners, type \dt.
- To list all of the tables, views, and sequences in the database, type \z.
- To exit the *psql* program, type \q.

PostgreSQL-Data Types

A datatype specifies, what kind of data you want to store in the table field. While creating table, for each column, you have to use a datatype. There are different categories of data types in PostgreSQL discussed below for your ready reference:

Type	**Data Type**	**Description**
Numeric Types	smallint	2-byte small-range integer
	integer, int	A signed, fixed precision 4-byte
	bigint	stores whole numbers, large range 8 byte
	real	4-byte, single precision, floating-point number
	serial	4-byte auto incrementing integer
	double precision	8-byte, double precision, floating-point number
	numeric(m,d)	Where m is the total digits and d is the number of digits after the decimal.
Character Types	character(n), char(n)	Fixed n-length character strings.
	character varying(n), varchar(n)	A variable length character string with limit.
	text	A variable length character string of unlimited length.
Monetary Types	money	currency amount,8 bytes
Boolean type	boolean	It specifies the state of true or false,1 byte.
Date/Time Type	date	date (no time of day),4 byte.
	time	time of day (no date),8 byte
	time with time zone	times of day only, with time zone,12 bytes
	bit(n)	Fixed-length bit string Where n is the length of the bit string.
	varbit(n) bit varying(n)	Variable-length bit string, where n is the length of the bit string.

Assignment no.5

Data Definition Query (Create)

Objective: To create simple tables, with only the primary key constraint (as a table level constraint & as a field level constraint) (include all data types)

A table is a database object that holds data. A table must have unique name, via which it can be referred. A table is made up of columns. Each column in the table must be given a unique name within that table. Each column will also have size a data-type and an optional constraint.

Syntax for **table creation** :

Create tablename(attribute list);

Attribute list : ([attribute name data type optional constraint] , ……….. .)

Create the following tables with primary key constraint

1. Create table emp (eno integer primary key, enamevarchar[50] , salary float);
2. Create table books(id integer UNIQUE, title text NOT NULL, author_idinteger,sub_idinteger,CONSTRAINTbooks_id_pkey PRIMARY KEY(id));
3. Create table sales_order(order_no char[10] PRIMARY KEY, order_date date, salesman_no integer);
4. Create table client_master (client_no integer CONSTRAINT p_client PRIMARY KEY, name varchar[50], addr text, bal_due integer);
5. Create table inventory(inv_no integer PRIMARY KEY, in_stock Boolean);
6. create table sales_order1(order_no char[10], product_no char[10],qty_orderedinteger,product_rate numeric(8,2),PRIMARY KEY(order_no,product_no));

SET A

1. Create a table with following details

Table Name	PLAYER		
Columns	Column Name	Column Data Type	Constraints
1	player_id	Integer	Primary key
2	Name	varchar (50)	
3	Birth_date	date	
4	Birth_place	varchar(100)	
Table level constraint			

2. Create a table with following details

Table Name	Student		
Columns	Column Name	Column Data Type	Constraints
1	Roll_no	integer	
2	Class	varchar (20)	
3	Weight	numeric (6,2)	
4	Height	numeric (6,2)	
Table level constraint	Roll_no and class as primary key		

3. Create a table with details as given below

Table Name	Project		
Columns	Column Name	Column Data Type	Constraints
1	project_id	integer	Primary key
2	Project_name	varchar (20)	
3	Project_ description	text	
4	Status	boolean	
Table level constraint			

4. Create a table with details as given below

Table Name	Donor		
Columns	Column Name	Column Data Type	Constraints
1	Donor_no	integer	Primary key
2	Donor_name	varchar (50)	
3	Blood_group	Char(6)	
4	Last_date	date	
Table level constraint			

Set B
Create table for the information given below by choosing appropriate data types and also specifying proper primary key constraint on fields which are underlined

1. Property (property_id, property_desc , area, rate, agri_status)
2. Actor (actor_id, Actor_name, birth_date)
3. Movie (movie-no, name, release-year)
4. Hospital (hno,hname,hcity)

Set C
Create table for the information given below by choosing appropriate data types and also specifying proper primary key constraint on fields which are underlined

1. Table ______ (________, ____________ , _____, ______, Primary key : ______________)

Instructor should fill in the blanks with appropriate values

Assignment Evaluation

0: Not Done [ ]   1: Incomplete [ ]   2: Late Complete [ ]

3: Needs Improvement [ ]   4: Complete [ ]   5: WellDone [ ]

Signature of Teacher

Assignment No.6

Objective: To create one or more tables with Check constraint , Unique constraint, Not null constraint , in addition to the first two constraints (PK & FK)

Constraints can be defined as either of the following:

```
CREATE TABLE table_name
(
column_name1TYPE column_constraint,
column_name2 type column constraint,
table_constrainttable_constraint
);
```

The following are the commonly used column constraints in PostgreSQL:

- NOT NULL – the value of the column cannot be NULL.
- UNIQUE – the value of the column must be unique across the whole table. However, the column can have many NULL values because PostgreSQL treats each NULL value to be unique. Notice that SQL standard only allows one NULL value in the column that has the UNIQUE constraint.
- PRIMARY KEY – this constraint is the combination of NOT NULL and UNIQUE constraints. You can define one column as PRIMARY KEY by using column-level constraint. In case the primary key contains multiple columns, you must use the table-level constraint.
- CHECK – enables to check a condition when you insert or update data. For example, the values in the price column of the product table must be positive values.
- REFERENCES – constrains the value of the column that exists in a column in another table. You use REFERENCES to define the foreign key constraint.

The following are the commonly used table constraints in PostgreSQL:

The table constraints are similar to column constraints except that they are applied to the entire table rather than to an individual column.

The following are the table constraints:

- UNIQUE (column_list)– to force the value stored in the columns listed inside the parentheses to be unique.
- PRIMARY KEY(column_list) – to define the primary key that consists of multiple columns.
- CHECK (condition) – to check a condition when inserting or updating data.
- REFERENCES– to constrain the value stored in the column that must exist in a column in another table.

Syntax for constraints

1. **Null constraint**
   Use of null constraint: Specifies that the column can contain null values
   Use of not null constraint: Specifies that the column can not contain null values
   Ex.: Create table client_master (client_no integer not null, name char(10) not null, addr varchar(30) null, bal_due numeric);

2. **Unique contarint**
   Use: forces the column to have unique value.
   Ex.: Create table client_master (client_no integer not null, name char(10) not null, addr varchar(30) null, bal_due numeric, ph_no integer unique);

3. **Check constraint**
   Use : Specifies a condition that each row in the table must satisfy.Condition is specified as a logical expression that evaluates either true or false.
   Ex. Create table client_master (client_no varchar CHECK(client_no like 'C%'), name char(10) check (name=upper(name)), addr varchar(30) null, bal_due numeric, ph_no integer unique);

Set A

1. Create a table with details as given below

Table Name	Machine		
Columns	Column Name	Column Data Type	Constraints
1	Machine_id	integer	Primary key
2	Machine_name	varchar (50)	NOT NULL, uppercase
3	Machine_type	varchar(10)	Type in ('drilling', 'milling', 'lathe', 'turning', 'grinding')
4	Machine_price	float	Greater than zero
5	Machine_cost	float	
Table level constraint	Machine_cost less than Machine_price		

2. Create a table with details as given below

Table Name	Employee		
Columns	Column Name	Column Data Type	Constraints
1	Employee_id	integer	Primary key
2	Employee_name	varchar (50)	NOT NULL, uppercase
3	Employee_desg	varchar(10)	Designation in ('Manager', 'staff', 'worker')
4	Employee_sal	float	Greater than zero
5	Employee_uid	text	Unique
Table level constraint	Employee_uid not equal to Employee_id		

Set B

1. Create a table with details as given below

Table Name	Student		
Columns	Column Name	Column Data Type	Constraints
1	Stud_id	integer	Primary key
2	Stud _name	varchar (50)	NOT NULL, uppercase
3	Stud _Class	varchar(10)	Class in ('FY', 'SY', 'TY')
4	Stud _Marks	float	Greater than zero
5	Stud _uid	text	Unique
Table level constraint	Stud_uid not equal to Stud_id		

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: WellDone [ ]

Signature of Teacher

Assignment No.7

1. **Data Definition Queries (Alter)** 2. **Data Definition Queries (Drop)**

Objective: To drop a table from the database, to alter the schema of a table in the Database.

1. **Alter Statement:** Alter table command is use to modify the structure of the table .

Syntax:
ALTER TABLE table_name action;

PostgreSQL provides many actions that allow you to:

- Add a column, drop a column, rename a column, or change a column's data type.
- Set a default value for the column.
- Add a CHECK constraint to a column.
- Rename a table.

The following illustrates the ALTER TABLE statement variants.

1. To **add a new column to a table**, you use **ALTER TABLE ADD COLUMN** statement:

   ALTER TABLE table_name ADD COLUMN new_column_name TYPE;

2. To **remove an existing column**, you use **ALTER TABLE DROP COLUMN** statement:

   ALTER TABLE table_name DROP COLUMN column_name;

3. To **rename an existing column**, you use the ALTER TABLE RENAME COLUMN TO statement:

   ALTER TABLE table_name RENAME COLUMN column_name TO new_column_name;

4. To **change a default value** of the column, you use ALTER TABLE ALTER COLUMN SET DEFAULT or DROP DEFAULT:

   ALTER TABLE table_name ALTER COLUMN [SET DEFAULT value | DROP DEFAULT]

5. To **change the NOT NULL constraint**, you use ALTER TABLE ALTER COLUMN statement:

   ALTER TABLE table_name ALTER COLUMN [SET NOT NULL| DROP NOT NULL]

6. To **add a CHECK constraint**, you use ALTER TABLE ADD CHECK statement:

   ALTER TABLE table_name ADD CHECK expression;

7. To **add a constraint**, you use ALTER TABLE ADD CONSTRAINT statement:

   ALTER TABLE table_name ADD CONSTRAINT constraint_name constraint_definition

8. To **rename a table** you use ALTER TABLE RENAME TO statement:

```
ALTER TABLE table_name RENAME TO new_table_name;
```

2. Drop Statement:

Use : Deletes an object (table/view/constraint) schema from the database.
Syntax: drop table table_name;
Example: drop table employee;

Set A

Create the table given below. Assume appropriate data types for attributes. Modify the table, as per the alter statements given below. Type \d <table name> and write the output.
Supplier_master(supplier_no, supplier_name,city,phone-no,amount)

1. Alter table supplier_master add primary key (supplier_no);
2. Alter table supplier_master add constraint city-check check city in('pune', 'mumbai', 'calcutta');
3. alter table supplier_master drop phone-no;
4. alter table supplier_master modify (supplier_namevarchar(50));
5. alter table supplier_master drop constraint city-check;
6. drop table supplier_master;

Set B

1. Remove employee table from your database. Create table employee(eno, ename, sal). Add designation column in the employee table with values restricted to a set of values.
2. Remove student table from your database.
Create table student(student_no, sname,date_of_birth).
Add new column into student relation named address as a text data type with NOT NULL integrity constraint and a column phone of data type integer.
3. Consider the project relation created in the assignment 12. Add a constraint that the project name should always start with the letter 'R'
4. Consider the relation hospital created in assignment 12. Add a column hbudget of type int , with the constraint that budget of any hospital should always > 50000.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Teacher

Assignment No.8

Data Manipulation Queries (Insert, Delete, Update)

Objective: To insert / update / delete records using tables created in previous Assignments. (Use simple forms of insert / update / delete statements)

INSERT syntax

```
INSERT INTO table_name (column1, column2 …) VALUES (value1, value2 …);
```

First, you specify the name of the table that you want to insert a new row after the INSERT INTO clause, followed by a comma-separated column list.

Second, you list a comma-separated value list after the VALUES clause. The value list must be in the same order as the columns list specified after the table name.

To add multiple rows into a table at a time, use the following syntax:

INSERT INTO table (column1, column2, …) VALUES (value1, value2, …),(value1, value2, …) ,…;
You just need to add additional comma-separated value lists after the first list, each value in the list is separated by a comma (,).

To insert data that comes from another table, use the INSERT INTO SELECT statement as follows:

INSERT INTO table(value1,value2,...) SELECT column1,column2,... FROM another_table WHERE condition;

The WHERE clause is used to filter rows that allow you to insert partial data from the another_table into the table.

Set A

Consider the tables created in previous assignments .Type and execute the below statements for these tables. Write the output of each statement & justify your answer

1. INSERT INTO sales_order(s_order_no,s_order_date,client_no) VALUES (‘A2100’, now() ,’C40014’);
2. INSERT INTO client_master values(‘A2100’,’NAVEEN’,’Natraj apt’, ’pune_nagar road’,’pune’,40014);
3. Insert into client_master values (‘A2100’,’NAVEEN’,NULL,’pune_nagar road’,’pune’,40014);
4. UPDATE emp SET netsal= net_sal_basic_sal*0.15;
5. UPDATE student
   SET name=’SONALI’,address=’Jayraj apartment’ WHERE stud_id=104 ;
6. DELETE from emp;
7. DELETE from emp WHERE net_sal<1000;

Set B

1. Create the following tables (primary keys are underlined.). Property(<u>pno</u> ,description, area) Owner(<u>oname,</u> address,phone)

An owner can have one or more properties, but a property belongs to exactly one owner . Create the relations accordingly ,so that the relationship is handled properly and the relations are in normalized form (3NF).

a) Insert two records into owner table.
b) insert 2 property records for each owner .
c) Update phone no of "Mr. Nene" to 9890278008
d) Delete all properties from "pune" owned by " Mr. Joshi"

2. Create the following tables (primary keys are underlined.).
Emp(eno,ename,designation,sal) Dept(dno,dname,dloc)
There exists a one-to-many relationship between emp & dept.Create the Relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF).
a) Insert 5 records into department table
b) Insert 2 employee records for each department.
c) increase salary of "managers" by 15%;
d) delete all employees of deparment 30;
e) delete all employees who are working as a "clerk"
f) change location of department 20 to 'KOLKATA'.

3. Create the following tables (primary keys are underlined.)
Sales_order(s_order_no,s_order_date)
Client(client_no, name, address)
A client can give one or more sales_orders , but a sales_order belongs to exactly one client. Create the Relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF).
a) insert 2 client records into client table
b) insert 3 sales records for each clientchange order date of client_no 'C004' to 12/4/08
c) delete all sale records having order date before 10th feb. 08
d) delete the record of client "joshi"

Set C
Create the following tables(Primary keys are underlined)
Machine (mno, name, mtype, mcost)
Part (pno, pname, pdesc)
Constraints : Primary Key constraints, machine name not null, foreign key
Machine & Parts are related with one-to-many relationship.
Create the relations accordingly,so that the relationship is handled properly and the relations are in normalized form(3NF) and insert 5 records into each table.

Solve the following queries:
a) Increase the cost of machine by 10%
b) List all parts whose machine cost is greater than 10001.

Assignment Evaluatio

0: Not Done [ ]	1: Incomplete [ ]	2:Late Complete[]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No.9

1.Simple queries **2.Aggregate queries**

Objective: To understand & get a Hands-on practice on Select statement

What is an aggregate function?

An aggregate function produced a single result for an entire group or table.

Aggregate functions are used to produce summarized results. They operate on sets of rows. They return results based on groups of rows. By default, all rows in a table are treated as one group. The GROUP BY clause of the select statement is used to divide rows into smaller groups.

List of aggregate functions

Name	**Description**
COUNT	This function returns the number or rows or non NULL values for a column
SUM	This function returns the sum of a selected column.
MAX	This function returns the largest value of a specific column.
MIN	This function returns the smallest value of a specific column.
AVG	This function returns the average value for a specific column.

Syntax

aggregate_name (expression [,...] [order_by_clause])

OR

aggregate_name (ALL expression [,...] [order_by_clause])

OR

aggregate_name (DISTINCT expression [,...] [order_by_clause])

OR

aggregate_name (*)

Set A

Create a table employee with attributes empno, name, address, salary and deptno. Insert atleast 10 records into the same. Execute each query

Execute following select queries .

1. Select * from emp;
2. Select empno, name from emp;
3. Select distinct deptno from emp;
4. Select * from emp where deptno =____;
5. Select * from emp where address = ‘pune’ and sal>______;
6. Select * from emp where address = ‘pune and salary between_______and_____;
7. Select * from emp where name like ‘---%’
8. Select * from emp where name like ‘%and%’

9. Select * from emp where salary is null;
10. Select * from emp order by eno;
11. Select * from emp order by deptno, enodesc;
12. Select deptno as department, sum(salary) as total from emp group by deptno order by deptno;
13. Select deptno as department , count(eno) as total_emp from emp group by deptno having count(eno) >______order by deptno;
14. select avg(salary) from emp;
15. select max(salary),deptno from emp group by deptno having max(sal) >___________;
16. select deptno, min(salary) from emp order by deptno;
17. update emp set salary = salary + 0.5*salary where deptno = (select deptno from department where dname = 'finance');
18. update emp set deptno = (select deptno from department where dname = 'finance')
Where deptno = (select deptno from department where dname = 'inventory');
19. insert into emp_backup(eno,ename) values(select eno,ename from emp);
20. delete from emp where deptno = (select deptno from department where dname='production');

Set B

Prerequisite : Students should know the normalization concept
Consider the relations

Person (pnumber, pname, birthdate, income),
Area (aname,area_type).
An area can have one or more person living in it , but a person belongs to exactly one area.
The attribute 'area_type' can have values as either urban or rural.
Create the Relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF).
Assume appropriate data types for all the attributes. Add any new attributes as required, depending on the queries. Insert sufficient number of records in the relations / tables with appropriate values as suggested by some of the queries**.**
Write select queries for following and execute them.

1. List the names of all people living in '_______'area.
2. List details of all people whose names start with the alphabet '_' & contains maximum _ alphabets in it.
3. List the names of all people whose birthday falls in the month of_______.
4. Give the count of people who are born on '_______'
5. Give the count of people whose income is below_______.
6. List names of all people whose income is between_______and______;
7. List the names of people with average income
8. List the sum of incomes of people living in '____'
9. List the names of the areas having people with maximum income (duplicate areas must be omitted in the result)
10. Give the count of people in each area
11. List the details of people living in '____' and having income greater than______;
12. List the details Of people, sorted by person number
13. List the details of people, sorted by area, person name
14. List the minimum income of people.
15. Transfer all people living in 'pune' to 'mumbai'.
16. delete information of all people staying in 'urban' area

Set C
Create the following tables(Primary keys are underlined):

Sailors(<u>sid</u>,sname,rate,age)
Boats(<u>bid</u>,bname,colour)
Reserves(sid,bid,date)

Sailors and boats are related many to many.
Create the relations accordingly,so that the relationship is handled properly and the relations are in normalized form(3NF) and insert 5 records into each table.
Draw ER diagram for given relational schema and show normalization.
Solve the following quesries:
a)Find all the sailors with a rating above 8.
b)Find the ages of sailors whose name begins and ends with 'P'.
c)Find name of sailors who have reserved red and green boats.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2:Late Complete[]
3: Needs Improvement [	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No.10
Queries with set operations

Objective: To understand & get a Hands-on practice using set operations (union ,intersect and except) with select statement.

1. **Union**
   Use: Returns the union of two sets of values, eliminating duplicates.
   Syntax: <select query> Union<select query>
   Ex.: Select cname from depositor Union Select cname from borrower;
2. **Union all**
   Use: returns union of two sets of values ,retaining all duplicates
   Syntax: <select query> Union all<select query>
   Ex.: Select cname from depositor Union allSelect cname from borrower;

3. **Intersect**
   Use:returns the intersection of two sets of values ,eliminating duplicates
   Syntax: <select query> intersect<select query>
   Ex.: Select cname from depositor intersect Select cname from borrower;

4. **Intersect all**
   Use: returns intersection of two sets of values ,retaining all duplicates
   Syntax: <select query> intersect all<select query>
   Ex.: Select cname from depositor intersect all Select cname from borrower;

5. **Except**
   Use: returns the difference between two sets of values.i.e returns all values of set1 not contained in set2,eliminates duplicates
   Syntax: <select query> except<select query>
   Ex.: Select cname from depositor **Except** Select cname from borrower;

6. **Except all**
   Use: returns the difference between two sets of values.i.e returns all values of set1 not contained in set2, retains all duplicates
   Syntax: <select query> except all<select query>
   Ex.: Select cname from depositor **Except** Select cname from borrower;

Note: To use the INTERSECT operator, the columns that appear in the SELECT statements must follow the rules below:

1. The number of columns and their order in the SELECT clauses must the be the same.
2. The data types of the columns must be compatible.

Set A

Consider the following relations, non-teaching, teaching, department.

One deparment can have one or more teaching & non-teaching staff, but a teaching or non-teaching staff belongs to exactly one department. Hence dno is a foreign key in the both the relations. Create these relations in your database .

Non-teaching (empnoint primary key, name varchar(20), address varchar(20), salary int,dno references department)
Teaching(empnoint primary key, name varchar(20), address varchar(20), salary int,dno references department)

Department (dnoint primary key,dname)

- insert at least 10 records into both the relations.
- type the following select queries & write the output and the business task performed by each query

1. Select empno from non-teaching union select empno from teaching;
2. Select empno from non-teaching union all select empno from teaching;
3. Select name from non-teaching intersect select name from teaching;
4. Select name from non-teaching intersect all select name from teaching;
5. Select name from non-teaching except select name from teaching;

6. Select name from non-teaching except all select name from teaching

Set B

Create the following relations, for an investment firm

emp(emp-id ,emp-name, address, bdate)

Investor(inv-name , inv-no, inv-date, inv-amt)

An employee may invest in one or more investments, hence he can be an investor.But an investor need not be an employee of the firm.

Create the Relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF).

Assume appropriate data types for the attributes. Add any new attributes , as required by the queries. Insert sufficient number of records in the relations / tables with appropriate values as suggested by some of the queries**.**

Write the following queries & execute them.

1. List the distinct names of customers who are either employees, or investors or both.
2. List the names of customers who are either employees , or investors or both.
3. List the names of employees who are also investors.
4. List the names of employees who are not investors

Set C

Employee (emp_no, emp_name, address, city, birth_date, designation,salary)

Project (project_no, project_name, status)

Department (Dept_no, dept_name, location)

Constraints: Employee designation is either 'manager', 'staff' , 'worker'.

There exists a one-to-many relationship between Department and Employee. Many employees can work on many projects controlled by a department. Create the relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF) and insert 5 records into each table.

Solve the following queries:

a) Find the details of employee who is having highest salary.
b) Delete all employees of department 20.
c) List the names and salary of employees sorted by their salary.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2:Late complete[]
3: Needs Improvement [	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No.11

Queries &sub-queries, with joining of tables

To understand & practice session on nested queries & sub-queries using join operations.

Sub query:

A sub-query is a select-from-where expression that is nested within another query.

Set membership	the 'in' & 'not in' connectivity tests for set membership & absence of set membership respectively.
Set comparison	the < some, > some, <= some, >= some, = some, <> some are the constructs allowed for comparison. = some is same as the 'in' connectivity. <> some is not the same as the 'not n'i connectivity. Similarly sql also provides < all, >all, <=all, >= all, <> all comparisons. <>all is same as the 'not in' construct.
Set cardinality	The 'exists' construct returns the value true if the argument subquery is nonempty. We can test for the non-existence of tuples in a subquery by using the 'not exists' construct. The 'not exists ' construct can also be used to simulate the set containment operation (the super set). We can write "relation A contains relation B" as "not exists (B except A)".

Set A

Create the following relation in your database (primary keys underlined)
Employee(ename, street, city)
Works(ename, company-name, salary)
Company(company-name, city)
Manages(ename, manager-name)

An employee can work in one or more companies, a company can have one or more employees working in it. Hence the relation 'works' with keyattributes as ename, company-name.

An employee manages one or more employees, but an employee is managed by exactly one employee (a recursive relationship), hence the relation 'manages' with key ename.

Insert sufficient number of records in the relations / tables with appropriate values as suggested by some of the queries**.**

Type the following queries , execute them and give the business task performed by each query

1. select ename from works w where salary >= all (select max(salary) from works));
2. select ename form works w where salary = (select max(salary) from works w1 where w1.company-name = w.company-name));
3. select manager-name from manages where manager-name in(select ename from works where company-name = "______");
4. select manager-name from manages where manager-name not in(select ename from works where company-name = "________");
5. select ename from works w where salary > some (select salary from works where company-name not in (select company-name from company where city = "_____"));
6. select ename from employee e where city = (select city from employee e1 , manages m where m.ename = e.ename and m.manager-name = e1.ename);
7. select * from employee where ename in (select manager-name from manages)

8 select city count(*) from employee group by city having count(*) >= all (select count(*) from

employee group by city)
9. select ename from works w where salary <> all (select salary from works where ename<>w.ename);
10. select company-name, sum(salary) from works w group by company-name having sum(sal) >= all (select sum(sal) from works group by company-name)
11. select ename from employee e where city in('_____','_____');
12. select ename from employee e where city = (select city from company c, works w where w.ename = e.name and c.company-name = w.company-name);

Set B

Create the following relations :

Emp(<u>eno</u>,name,dno,salary)
Project(<u>pno</u>,pname,control-dno,budget)

Each employee can work on one or more projects, and a project can have many employees working in it. The number of hours worked on each project , by an employee also needs to be stored.
Create the Relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF).
Assume appropriate data types for the attributes. Add any new attributes , new relations as required by the queries.
Insert sufficient number of records in the relations / tables with appropriate values as suggested by some of the queries**.**
Write the queries for following business tasks & execute them.

1. list the names of departments that controls projects whose budget is greater than_____.
2.list the names of projects, controlled by department No , whose budget is greater than atleast one project controlled by department No .
3. list the details of the projects with second maximum budget

4. list the details of the projects with third maximum budget.
5. list the names of employees, working on some projects that employee number is working.
6. list the names of employees who do not work on any project that employee number works on
7. list the names of employees who do not work on any project controlled by '______' department
8. list the names of projects along with the controlling department name, for those projects which has atleast employees working on it.
9. list the names of employees who is worked for more than 10 hrs on atleast one project controlled by '______' dept.
10. list the names of employees , who are males , and earning the maximum salary in their department.
11. list the names of employees who work in the same department as '______'.
12. list the names of employees who do not live in_______or_______**.**

Set C
Execute following queries on the relations mentioned in above case study
1. list the names of projects along with the controlling department name, for those projects which has atleast   employees working on it.
2. list the names of employees who is worked for more than 10 hrs on atleast one project controlled by '______' dept.
3. list the names of employees , who are males , and earning the maximum salary in their department.
4. list the names of employees who work in the same department as '______'.
5. list the names of employees who do not live in_______or________.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2:Late Complete[]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No.12

Queries &sub queries, with joining of table

Execute the following queries on the table created in previous assignments.

Set A

Project-Employee database

Consider the database maintained by a company which stores the details of the projects assigned to the employees.
Following are the tables:

PROJECT (PNO INTEGER, P_NAME CHAR(30), PTYPE CHAR(20),DURATION INTEGER)
EMPLOYEE (ENO INTEGER, E_NAME CHAR (20), QUALIFICATION CHAR (15), JOINDATE DATE)

The relationship is as follows:

PROJECT - EMPLOYEE: M-M Relationship , with descriptive attributes as start_date (date), no_of_hours_worked (integer).

Solve the Queries

1. Find the names of the employees starting with 'A'.
2. Find the details of employees working on the project "System".
3. Find the employee numbers of the employees, who do not work on project "Robotics".
4. Get employee number of the employee, who works on at least one project that employee number '2000' works on.
5. List the names of the first three employees in alphabetical order.
6. Find the names of the employees whose duration is more than three years.

Set B

Bank database

Consider the following database maintained by a Bank. The Bank maintains information about its branches, customers and their loan applications.

Following are the tables:

BRANCH (bid integer, brname char (30), brcity char (10))
CUSTOMER (cno integer, cname char (20), caddr char (35), city (20))
LOAN_APPLICATION (lno integer, lamtrequired money, lamtapproved money, l_date date)

The relationship is as follows:

BRANCH, CUSTOMER, LOAN_APPLICATION are related with ternary relationship.
TERNARY (bid integer, cno integer, lno integer).

Solve the Queries

1. Find the names of the customers for the "Aundh" branch.
2. List the names of the customers who have received loan less than their requirement.
3. Find the maximum loan amount approved.
4. Find out the total loan amount sanctioned by "Deccan "branch.
5. Count the number of loan applications received by "M.G.ROAD" branch.
6. List the names of the customer along with the branch names who have applied for loan in the month of September.

Set C
Student- Teacher database
Consider the following database maintained by a school. The school maintains information about students and the teachers. It also gives information of the subject taught by the teacher.

Following are the tables:
STUDENT (sno integer, s_name char(30), s_class char(10), s_addr char(50))
TEACHER (tno integer, t_name char (20), qualification char(15),experience integer)

The relationship is as follows:
STUDENT-TEACHER: M-M with descriptive attribute SUBJECT.

Solve the queries
1. Find the minimum experienced teacher.
2. Find the number of teachers having qualification "Ph. D.".
3. List the names of the students to whom "Mr. Patil" is teaching along with the subjects he is teaching to them.
4. Find the subjects taught by each teacher.
5. List the names of the teachers who are teaching to a student named "Suresh".
6. List the names of all teachers along with the total number of students they are teaching.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2:Late Complete[]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor

Assignment No.13
Case studies

Set A

Business trip database

Consider the business trip database that keeps track of the business trips of salesman in an office.

Following are the tables:

SALESMAN (sno integer, s_name char (30), start_year year, deptno varchar (10))
TRIP (tno integer, from_city char (20), to_city char (20), departure_date date, return date)
DEPT (deptno varchar (10), dept_name char(20)) ,expense (eid integer, amount money)
The relationship is as follows
DEPT-SALESMAN 1 TO M
SALESMAN - TRIP 1 TO M
TRIP - EXPENSE 1 TO 1

Execute the following queries

1. Increase the expenses of all the trips by Rs. 5000.
2. Give the details for trips that exceed Rs. 10,000 in expenses.
3. List the salesman numbers and names of the salesmen who made trips to Calcutta.
4. Delete all the trips made by department "computer" having expenses more than Rs.15000.
5. Find the departments from which the salesmen have done highest number of trips.
6. Find the total expenses incurred by the salesman "Mr. Patil".

Set B

Warehouse Database

CITIES (city char (20), state char(20))
WAREHOUSES (wid integer, wname char (30), location char(20))
STORES (sid integer,store_name char(20), location_city char(20))
ITEMS (itemno integer, description text, weight decimal(5,2), cost decimal(5,2))
CUSTOMER(cno integer, cname char(50),addr varchar(50), c_city char(20))
ORDERS(o_no int, o_date date)

The relationship is as follows

CITIES-WAREHOUSES 1 TO M
WAREHOUSES - STORES 1 TO M
CUSTOMER – ORDERS 1 TO M
ITEMS – ORDERS M TO M relationship with descriptive attribute ordered_quantity
STORES-ITEMS M TO M RELATION with descriptive attribute quantity

Solve the following queries.

1. Find the item that has minimum weight.
2. Find the different warehouses in "Pune".
3. Find the details of items ordered by a customer "Mr. Patil".
4. Find a Warehouse which has maximum stores.
5. Find an item which is ordered for minimum number of times.
6. Find the details orders given by each customer.

Set C
movie database
movies(m_name, release_year, budget)
actor(a_name, role, charges, a_address)
producer(producer_id, name, p_address)
each actor has acted in one or more movies. each producer has produced many movies and each movie can be produced by more than one producers. each movie has one or more actors acting in it, in different roles.
create the relations accordingly, so that the relationship is handled properly and the relations are in normalized form(3nf).
insert sufficient number of appropriate records.
solve the queries:
1. list the names of actors who have acted in at least one movie, in which '_______' has acted.
2. list the names of the movies with the highest budget.
3. list the names of actors who have acted in the maximum number of movies.
4. list the names of movies, produced by more than one producer.
5. list the names of actors who are given with the maximum charges for their movie.
6. list the names of producers who produce the same movie as '________'.
7. list the names of actors who do not live in_________or_________city.

Assignment Evaluation

0: Not Done [ ]	1: Incomplete [ ]	2:Late Complete[]
3: Needs Improvement [ ]	4: Complete [ ]	5: Well Done [ ]

Signature of Instructor