Anekant Education Society's

Tuljaram Chaturchand College of Arts, Science & Commerce, Baramati

(AUTONOMOUS)

Affiliated

Savitribai Phule Pune University, Pune

Department Of Computer Science

F.Y.B.Sc.(Computer Science)

Lab Course I (CSCO-1103) – Basics of 'C'

Lab Course II (CSCO -1104) – DBMS - I

(Semester I)

20 - 20

Name_____

Roll No: _____ Division_____

Anekant Education Society's

Tuljaram Chaturchand College of

Arts, Science & Commerce Baramati(Autonomous)

DEPARTMENT OF COMPUTER SCIENCE

Computer Practical Journal

Paper III: CSCO – 1103 Lab Course on Basic C

- CERTIFICATE -

University Seat No. Date :

This is to certify that Mr./ Miss. / Smt. _____

_____ has satisfactorily completed the course in practical as prescribed by the University of Pune for the **F.Y.B.Sc.(Computer Science-Lab Course I** in the Year 2019 – 2020

In-Charge (Practical) Internal Examiner External Examiner Head Computer Science

Index

Sr. No.	Assignment Name	Date	Remark
1	To demonstrate the use of data types, simple operators and expressions.		
2	Assignment to demonstrate decision making statements (if and if-else, nested structures)		
3	Assignment to demonstrate decision making statements (switch - case)		
4	Assignment to demonstrate use of simple loops		
5	Assignment to demonstrate use of nested loops		
6	Assignment to demonstrate menu driven programs.		
7	Assignment to demonstrate writing C programs in modular way (use of user defined functions)		
8	Assignment to demonstrate recursive functions.		
9	Assignment to demonstrate use of arrays (1-d arrays) and functions		
10	Assignment to demonstrate use of arrays (1-d arrays) and functions		

Assignment:1

To demonstrate the use of data types, simple operators and expressions

1. Data type Table

Data	Data Format	C Data Type	C Variable Declaration	Input Statement	Output Statement

quantity
month
credit-card number
	Numeric	int
		Short int
		long int
			int quantity;
			short month;
			long ccno;
				scanf("%d",&quantity);
				scanf("%d",&month);
				scanf("%ld", &ccno);
					printf("The quantity is %d", quantity);
					printf("The credit card number is %ld, ccno);

Price	Real	float double	float price; const double pi=3.141593;	scanf("%f",&price);	printf("The price is %5.2f", price);
Grade	character		char grade;	scanf("%c",&grade)	printf("The grade is %c",grade);

2. Expression Examples

Expression	C expression
Increment by a 3	a = a + 3
Decrement b by 1	b = b-1 or b--
2 a^2 + 5 b/2	2*a*a + 5*b/2
7/13(x- 5)	(float)7/13*(x-5)
5% of 56	(float)5/100*56
n is between 12 to 70	n>=12 && n<=70
¾r^2h	Pi*r*r*h
n is not divisible by 7	n % 7 != 0
n is even	n%2==0
ch is an alphabet	ch>='A' && ch<='Z' || ch>='a' && ch<='z'

Note: The operators in the above expressions will be executed according to precedence and associatively rules of operators.

Follow the following guidelines

a. Type the sample program save it pnr.c
Compile the program using gcc compiler
available in Linux(red hat) gcc pnr.c

A executable file a.out is created by the compiler in current directory. The program can be executed by typing name of the file as follows giving the path.
./a.out

Write Programs to solve the following problems

Set – A

1. Write a C program to display message “Hello ! Welcome To C” using printf function.
2. Write a C program to demonstrate declaration of variables and functions.
3. Write a C program for addition, substraction, multiplication and division.
4. Write a C program to accept principal sum, rate of interest and number of years. Compute simple interest.

Set – B

1. Write a C program to accept three dimensions length(l),breadth(b)and height(h) of a cuboid and print surface area and volume (Hint: Surface area = 2(lb+lh+bh),volume =lbh)
2. Write a C program to calculate area of circle, circumference of circle (2*PI*r), area of rectgangle.
3. Write a C program to accept marks of five subjects and display percentage.
4. Write a C program to accept empid , basic salary calculate total salary including tax , dearness allowance.
5. Write a C program to a cashier has currency notes of denomination 1, 5, 10. Accept the amount to be withdrawn from the user and print the total number of currency notes of each denomination the cashier will have to give.

---------- ---------- ----------

Signature of the Instructor **Remark** **Date**

Assignment:2

To demonstrate use of decision making statements such as if and if-else.

During problem solving, we come across situations when we have to choose one of the alternative paths depending upon the result of some condition. Condition is an expression evaluating to true or false. This is known as the Branching or decision-making statement. Several forms of If and else constructs are used in C to support decision-making.

1) if statements

```
if (condition)
{
Statement;
}
```

2) if…else

```
if(condition)
  {
        statement;
  } else
      {
   Statement;

      }
```

Write Programs to solve the following problems

Set – A

1. Write a C program to check whether number is positive and negative.
2. Write a C program to check whether number is even or odd.
3. Write a C program to check whether number is divisible by 5 and 7.
4. Write a C program to Check year is leap year or not .

Set – B

1. Write a C program to Check whether character is a digit or a character or other symbol Also check character is in lowercase or uppercase (ASCII value of digit is bet 48 to 58, a z is 97 to 123, A Z 65 to 96)
2. Write a C program to Accept marks of three subjects and find the total marks, average. Display class obtained
3. Write a C program to Calculate roots of quadratic equation .

---------- ---------- ----------

Signature of the Instructor **Remark** **Date**

Assignment:3

To demonstrate decision making statements (switch case)

The control statement that allows us to make a decision from the number of choices is called a switch-case statement. It is a multi-way decision making statement.

Switch syntax:

```
switch(expression)
{
  case value 1:statement 1;
   break;
  case value 2:statement 2;
  break;
  case value n:statement n;
  break;
   default: default statement;
   break;
 }
```

Write Programs to solve the following problems

Set – A

1. Write a C program to Display selected number or option
2. Write a c program Accept two integers and perform arithmetic operations

Set – B

1. Accept single digit and display it in words.
2. Write a c program having a menu with the following options and corresponding actions 1. Area of circle 2.Area of Rectangle 3. Area of triangle .

---------- ---------- ----------

Signature of the Instructor **Remark** **Date**

Assignment:4

To demonstrate use of simple loops.

We need to perform certain actions repeatedly for a fixed number of times or till some condition holds true. These repetitive operations are done using loop control statements. The types of loop structures supported in C are

1)while statement	**2)**do-while statement	**3)for statement**
while(condition)	do{	for(expr1;expr2;expr3)
{statement 1;	statement 1;	{
Statement 2;	Statement 2;	Statement1;
}	}while(condition);	Statement2;
		}

Write Programs to solve the following problems

Set – A

1. Write a C program to Accept a number and reverse it . (if input 978 o/p 879)
2. Write a C program to Accept a number & find sum of digits
3. Write a C program to Accept a number & check whether it is a Armstrong Number (sum of cube of its digit = number itself)
4. Write a c program to accept characters from the keyboard till the use enter EOF (ctrl+Z) and count the number of vowels ,spaces and line in the text.

Set – B

1. Write a C program to accept a number and display its digits in Words.
   (Input : 572 output: FIVE SEVEN TWO)
2. Write a C program to accept a number and calculate factorial of given number.
3. Write a C program to accept a number and check whether number is Perfect or not.
   (Hint : sum of factors = number itself)
4. Write a C program to find Fibonacci series i.e. 1 1 2 3 5 8….
   (Hint : sum of previous two numbers is the next term)
5. Write a C program to accept an integer and check if it is prime or not.

---------- ---------- ----------

Signature of the Instructor **Remark** **Date**

Assignment:5

To demonstrate use of nested loops

Nested loop means a loop that is contained within another loop. Nesting can be done upto any levels. However the inner loop has to be completely enclosed in the outer loop. No overlapping of loops is allowed.
let to nest any loop within another. For example, we can have a for loop inside a while or do while or a while loop inside a for.

Write C programs for the following problems.

Set – A

1. The Sample program 1 displays n lines of the following triangle. Type the program and execute it
for different values of n.

```
1
1   2
1   2   3
1   2   3   4
```

2. Modify the sample program 1 to display n lines of the Floyd's triangle as follows (here n=4).

```
1
2   3
4   5   6
7   8   9   10
```

3. Write a C program to calculate sum of first n terms of series 1+3+5+…….

4. Write a C program to calculate sum of first n terms of series X + 3x + 5x +7x + ……

Set – B

1. Write a C to accept a number x and integer n and calculate the sum of first n terms of series.
$1/x + 2/x^2 + 3/x^3 + \ldots$.
2. Write a C program to display all Armstrong numbers between 1 to 500.
3. Write a C program to Display following output :

```
A B C D
E F G
 H I
 J
```

4. Write a C program to display multiplication table from 2 to 9.

---------- ---------- ----------

Signature of the Instructor **Remark** **Date**

Assignment:6

To demonstrate menu driven programs and use of standard library functions

A function is a named sub-module of a program, which performs a specific, well- defined task. It can accept information from the calling function in the form of arguments and return only 1 value. C provides a rich library of standard functions. We will explore some such function libraries and use these functions in our programs.

ctype.h : contains function prototypes for performing various operations on characters.Some commonly used functions are listed below.

Function Name	Purpose	Example
Isalpha	Check whether a character is a alphabet	if (isalpha(ch))
Isalnum	Check whether a character is alphanumeric	if (isalnum(ch))
Isdigit	Check whether a character is a digit	if (isdigit(ch))
Isspace	Check whether a character is a space	if (isspace(ch))
Ispunct	Check whether a character is a punctuation Symbol	if (ispunct(ch))
Isupper	Check whether a character is uppercase alphabet	if (isupper(ch))
Islower	Check whether a character is lowercase alphabet	if (isupper(ch))
Toupper	Converts a character to uppercase	ch = toupper(ch)
Tolower	Converts a character to lowercase	ch = tolower(ch)

math.h : This contains function prototypes for performing various mathematical operations on numeric data. Some commonly used functions are listed below.

Function Name	Purpose	Example
Cos	Cosine	a*a+b*b – 2*a*b*cos(abangle)
exp(double x)	exponential function, computes e^x	exp(x)
Log	natural logarithm	c= log(x)
log10	base-10 logarithm	y=log10(x)
pow(x,y)	compute a value taken to an exponent, x^y	y = 3*pow(x , 10)
Sin	Sine	z= sin(x) / x
Sqrt	square root	delta=sqrt(b*b – 4*a*c)

Set A

1. Write a menu driven program to perform the following operations on a character type variable.
    a. Check if it is an alphabet
    b. Check if it is a digit.
    c. Check if it is lowercase.
    d. Check if it is uppercase.
    e. Convert it to uppercase.

2. Write a menu driven program to perform the following operations till the user selects Exit. Accept appropriate data for each option. Use standard library functions from math.h
    i. Sin ii. Cosine iii. log iv. e^x
    v. Square Root vi. Exit

Set B

1. Accept two complex numbers from the user (real part, imaginary part).

2. Write a menu driven program to perform the following operations till the user selects Exit.
    i. Add ii. Substract iii. Multiply iv. Exit

---------- ---------- ----------

Signature of the Instructor **Remark** **Date**

Assignment:7

To demonstrate writing C programs in modular way (use of user defined functions

You have already used standard library functions. C allows to write and use user defined functions. Every program has a function named main. In main you can call some functions which in turn can call other functions.

The following table gives the syntax required to write and use functions

Sr. No	Actions involving functions	Syntax	Example
1.	Function declaration	returntype function(type arg1, type arg2 …);	void display(); int sum(int x, int y);

2.	Function definition	returntype function(type arg1, type arg2 …)
		{
		/* statements*/
		}
			float calcarea (float r)
			{
			float area = Pi *r*r
			; return area;
			}

3.	Function call	function(arguments); variable = function(arguments);	display(); ans = calcarea(radius);

Set A

1. Write a C program to accept two numbers and Find maximum number between two numbers using function.
2. Write a C program for swapping of two numbers using function.

Set B

1. Write a C program to Calculate factorial using function.
2. Write a C program to accept two nos.+,/,*,%using function.

---------- ---------- ----------

Signature of the Instructor **Remark** **Date**

Assignment: 8

To demonstrate Recursion

Recursion is a process by which a function calls itself either directly or indirectly. The points to be remembered while writing recursive functions

i. Each time the function is called recursively it must be closer to the solution.

ii. There must be some terminating condition, which will stop recursion.

iii. Usually the function contains an if –else branching statement where one branch makes recursive call while other branch has non-recursive terminating condition

Set A

1. Write a recursive C function to calculate the sum of digits of a number. Use this function in main to accept a number and print sum of its digits.
2. Write a recursive C function to calculate the GCD of two numbers. Use this function in main. The GCD is calculated as:
   gcd (a , b) = a if b = 0
   gcd (b, a mod b) otherwise

Set B

1. Write a recursive C function to calculate x^y. (Do not use standard library function)
2. Write a recursive function to calculate the sum of digits of a number till you get a single digit number. Example: 961 -> 16 -> 5. (Note: Do not use a loop)

| --- | --- | --- |
| **Signature of the Instructor** | **Remark** | **Date** |

Assignment:9

To demonstrate use of 1-D arrays and functions.

An array is a collection of data items of the same data type referred to by a common name. Each element of the array is accessed by an index or subscript. Hence, it is also called a subscripted variable.

1. How to declare one-dimensional array Syntax:-Data-type arrary_name[size];

   Example: int mat1 [10];

Set A

1. Write a C program to Accept elements for an array & display elements of an Array.
2. Write a C program to Display even numbers from an array

Set B

1. Write a C program to Find smallest & largest number from an Array
2. Write a function, which accepts an integer array and an integer as parameters and counts the occurrences of the number in the array.

---------- ---------- ----------

Signature of the Instructor **Remark** **Date**

Assignment:10

To demonstrate use of 2-D arrays and functions.

How to declare two-dimensional array

Syntax - Data-type arrary_name[size][size];
Example:int mat1[10][10];

Accessing elements

Array_name [index1][index2] where index1is the row & index 2 is the column location of an elements in the array.

Set A

1. Write a C program to Accept elements for an array and sort elements of an array
2. Write a C program to Accept elements for an array and Merge two sorted arrays into a third array.
3. Write a C program for Addition of two matrices

Set B

1. Write a c program for multiplication of two matrices.
2. Write a C program to Transpose of a matrix .

---------- ---------- ----------

Signature of the Instructor **Remark** **Date**

Anekant Education Society's

Tuljaram Chaturchand College of

Arts, Science & Commerce Baramati(Autonomous)

DEPARTMENT OF COMPUTER SCIENCE

Computer Practical Journal

Paper IV: CSC – 1104 Lab Course on DBMS

- CERTIFICATE -

University Seat No. Date :

This is to certify that Mr./ Miss. / Smt. _____

_____ has satisfactorily completed the course in practical as prescribed by the University of Pune for the **F.Y.B.Sc.(Computer Science-Lab Course II** in the Year 2019 – 2020

In-Charge (Practical) Internal Examiner External Examiner Head Computer Science

Index

Sr.No.	Title of Experiment/ Practical	Date	Sign
1	Create simple tables , with only the primary key Constraint		
2	Create more than one table with integrity constraint		
3	Create more than one table, with referential integrity constraint.		
4	Drop a table from database, Alter the table.		
5	Insert/Update/Delete statements.		
6	Query for the tables using simple form of Select Statement		
7	Query solving for table operations(Aggregate function)		
8	Nested Query solving for table operations(Union, Intersect, Except)		
9	Nested Query solving for table operations(Set membership, Cardinality, Comparison)		
10	Small Case Studies.		

Assignment: 01

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

PostgreSQL(pronounced as **post-gress-Q-L**) is an open source relational database management system (DBMS) developed by a worldwide team of volunteers. PostgreSQL is not controlled by any corporation or other private entity and the source code is available free of charge. PostgreSQL runs on all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows. PostgreSQL supports a large part of the SQL standard and offers many modern features including the Complex SQL queries, SQL Sub-selects, Foreign keys, Trigger, Views, Transactions, Multiversion concurrency control (MVCC) etc.

PostgreSQL supports a wide variety of built-in data types, and it also provides an option to the users to add new data types to PostgreSQL. Table lists the data types officially supported by PostgreSQL. Most data types supported by PostgreSQL are directly derived from SQL standards.

Introduction to PostgreSQL CREATE DATABASE statement

- To create a new PostgreSQL database, you use CREATE DATABASE statement as shown below:
  CREATE DATABASE db_name;
- Once a database is no longer needed, you can delete it by using the DROP DATABASE statement.
  The following illustrates the syntax of the DROP DATABASE statement:
  DROP DATABASE [IF EXISTS] name;
- To delete a database:
  - Specify the name of the database that you want to delete after the DROP DATABASE clause.
  - Use IF EXISTS to prevent an error from removing a non-existent database. PostgreSQL will issue a notice instead.
- The DROP DATABASE statement deletes catalog entries and data directory permanently. This action cannot be undone so you have to use it with caution.

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. The following table contains PostgreSQL supported data types for your reference:

Category		Numeric	
Name	**Storage size**	**Description**	**Range**
Integer	4 Bytes	Small range integer	-32768 to 32768
Numeric	Variable	User specified precision,exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
smallint, int2	2-bytes	A signed integer	-32768 to +32767
Bigint/ int8	8 bytes	large-range integer	-9223372036854775808 to 9223372036854775807
Real/Float	4 Bytes	Variable presion,inexact	6 Decimal Digit precision
Serial	4Bytes	Autoincrement integer	1 to 2147483647
Category		**Character**	
Name		**Description**	
Character(n)		A fixed n-length character string	
Char(n)		A fixed n-length character string	
Character varying (n)		Variable with limit	
Varchar(n)		Variable length with limit	
Text		Variable character string	
Category		**Currency**	
Name	**Storage size**	**Description**	**Range**
Money	8 Bytes	Currency amount	-92233720368547758.08 to +9223372036854775 8.07
Category		**Binary**	
Name		**Description**	
bit(n)		Fixed-length bit string Where ***size*** is the length of the bit string.	
varbit(*size*) bit varying(*size*)		Variable-length bit string where ***size*** is the length of the bit string.	

Category		Date/Time	
Name	**Storage size**	**Description**	**Range**
timestamp [(p)] [without time zone]	8 bytes	both date and time (no time zone)	4713 BC to 294276 AD
timestamp [(p)] with time zone	8 bytes	both date and time, with time zone	4713 BC to 294276 AD
Date	4 bytes	date (no time of day)	4713 BC to 5874897 AD
time[(p)][without time zone]	8 bytes	time of day (no date)	00:00:00 to 24:00:00
time[(p)] with time zone	12 bytes	times of day only, with time zone	00:00:00+1459 to 24:00:00-1459
interval[fields][(p)]	12 bytes	time interval	-178000000years to 178000000 years

Category		Boolean
Name	**Storage size**	**Description**
Boolean,bool	1 byte	State true or false

Syntax for table creation :

Create table tablename (attribute list);

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

Constraints can be defined as either of the following :

Name	Description	Example
Column level	When data constraint is defined only with respect to one column & hence defined after the column definition, it is called as column level constraint.	Create table tablename (attribute1 datatype primary key, attribute2 datatype constraint constraint-name,……);
Table Level	When data constraint spans across multiple columns & hence defined after defining all the table columns when creating or altering a table structure, it is called as table level constraint	Create table tablename (attribute1 datatype , attribute2 datatype2 , ……, constraint pkey primary key (attribute1, atrribute2));

Set A:

1. Create a table with details as given below:

Table Name		Game	
Columns	**Column Name**	**Column Data Type**	**Constraints**
1	G_Code	Integer	Primary Key
2	G_name	Varchar(50)	
3	G_tournment	Date	
4	G_disc	Varchar(100)	
Table level Constraint			

2. Create a table with details as given below

Table Name		Student	
Columns	Column Name	Column Data Type	Constraints
1	Roll_no	Integer	Primary Key
2	Class	Varchar(20)	
3	Weight	Numeric(6,2)	
4	Height	Numeric(6,2)	
Table level Constraint		Roll_no & 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	P_disc	Text	
4	Status	Boolean	
Table level Constraint			

4.Create a table with details as given below:

Table Name		Donar	
Columns	**Column Name**	**Column Data Type**	**Constraints**
1	D_no	Integer	Primary Key
2	D_name	Varchar(20)	
3	Blood_grp	Char(6)	
4	Last_date	Date	
Table level Constraint			

Set B :

1. Create a table with details as given below:

Table Name		Game	
Columns	**Column Name**	**Column Data Type**	**Constraints**
1	G_no	Integer	Primary Key
2	G_name	Varchar(20)	
3	No_of_player	Integer	
4	Coach Name	Varchar(50)	
Table level Constraint			

2.Create a table with details as given below:

Table Name		Musician	
Columns	**Column Name**	**Column Data Type**	**Constraints**
1	M_no	Integer	
2	M_name	Varchar(50)	
3	Age	Interger	
4	M_address	Varchar(100)	
Table level Constraint		Primary Key	

---------- ---------- ----------

Signature of the Instructor **Remark** **Date**

Assignment: 02

To create more than one table, with referential integrity constraint, primary key constraint.

The following is the list of constraints that can be defined for enforcing the referential integrity constraint.

Constraint	Use	Syntax & Example
Primary key	Designates a column or combination of columns as primary key	PRIMARY KEY(columnname,columnname)
Foreign key	designates a column or grouping of columns as a foreign key with a table constraint	FOREIGN KEY (columnname,columnname)
References	Identifies the primary key that is referenced by a foreign key. If only parent table name is specified it automatically references primary key of parent table	columnname datatype REFERENCES tablename[columnname]
On delete Cascade	The referential integrity is maintained by automatically removing dependent foreign key values when primary key is deleted	Columnname datatype REFERENCES tablename [columnname] [ON DELETE CASCADE]
On update set null	If set, makes the foreign key column NULL, whenever there is a change in the primary key value of the referred Table	Constraint name foreign key column- name references referred- table(column-name) on update set null

Set A :

1. Create the following tables :

Table Name		Property	
Columns	**Column Name**	**Column Data Type**	**Constraints**
1	Pnumber	Integer	Primary key
2	Description	varchar (50)	Not null
3	Area	Char(10)	

Table Name		Owner	
Columns	**Column Name**	**Column Data Type**	**Constraints**
1	Owner-name	Varchar(50)	Primary key
2	Address	varchar (50)	
3	Phoneno	Integer	

Relationship is one-many between Owner & Property. Define reference keys.

2. Create the following tables :

Table Name		Hospital	
Columns	**Column Name**	**Column Data Type**	**Constraints**
1	Hno	Integer	Primary key
2	Name	varchar (50)	Not null
3	City	Char(10)	

Table Name		Doctor	
Columns	**Column Name**	**Column Data Type**	**Constraints**
1	Dno	Integer	Primary key
2	Dname	varchar (50)	
3	City	Char(10)	

Relationship is many-many between Hospital & Doctor.

3. Create the following tables:

Table Name		Patient	
Columns	**Column Name**	**Column Data Type**	**Constraints**
1	Pno	Integer	Primary key
2	Name	varchar (50)	Not null
3	Address	Varchar(50)	

Table Name		**Bed**	
Coloumn	**Coloumn Name**	**Coloumn Data Type**	**Constraint**
1	Bedno	Integer	Primary key
2	Roomno	Integer	
3	Discription	Varrchar(50)	

Relationship is one–to-one relationship between patient & bed.

Set B :

1 Create the following tables according mapping cardinality given below.

I. Country(Con-code, Name, Capital) Population(Pop-code, Population)

Country and Population are related with one to one relationship.

II. Employee (empno, empname, salary, comm.., desg)
Department (deptno, deptname, location)

Employee and Department are related with many to one relationship.

III. Game(Gno, Gname, No. of Player, Coachname, Captain)
Player(Pno, Pname)

Game and Player are related with many to many relationships.

---------- ---------- ----------

Signature of the Instructor **Remark** **Date**

Assignment: 03

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

The following is the list of additional integrity constraints.

Constraint	Use	Syntax & Example
Null	Specifies that the column can contain null values	Create table Client_Master (Client_no text Not Null, Name text Not Null, Addr text Null, Balance Integer);
Not Null	Specifies that the column can contain not null values	Create table client_Master (Client_no text Not Null, Name text Not Null);
Unique	Forces the column to have unique values	Create table client_Master (Client_no text Unique, Name text Not Null);
Check	Specifies a condition that each row in the table must satisfy.	Create table client_Master (Client_no text primary key, Name text, Constraint name_chk check(Name=upper(Name)));

Set A :

1. Create a table with details as given below

Table Name	Machine	
Column Name	**Data Type**	**Constraints**
Machine_id	Integer	Primary key
Machine_name	varchar (50)	NOT NULL, uppercase
Machine_type	varchar(10)	Type in (‘drilling’, ‘Milling’, ‘Lathe’, ‘Turning’, ‘Grinding’)
Machine_price	Float	Greater than Zero
Machine_cost	Float	
Table level constraint	Machine_cost less than Machine_price	

2.Create a table with details as given below

Table Name	Employee	
Column Name	**Data Type**	**Constraints**
Emp_id	Integer	Primary key
Emp_name	Varchar (50)	NOT NULL, uppercase
Emp_desg	Varchar(10)	Designation in('Manager', 'staff', worker')
Emp_sal	Float	Greater than zero
Emp_uid	Text	Unique
Table level constraint	Emp_uid not equal to Empl_id	

Set B :

Table Name		Department	
Columns	**Column Name**	**Column Data Type**	**Constraints**
1	Dept_id	Integer	Primary key
2	Dept_name	varchar (50)	NOT NULL, uppercase
3	Dept_type	varchar(20)	Type in ('drilling', 'milling', 'lathe', 'turning', 'grinding')
4	Dept_budget	Integer	Greater than zero
5	Dept_expenditure	Float	
Table level constraint		Dept_expenditure less than Dept_budget	

---------- ---------- ----------

Signature of the Instructor **Remark** **Date**

Assignment: 04

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

The Drop & Alter DDL statements:

Drop:-Deletes an object (table/view/constraint)schema from the database.

Drop object-type Object-name; e.g. Drop table employee;

Alter:- ALTER TABLE command of SQL is used to modify the structure of the table. It can be used for following purposes :

a) Adding new column b) Modifying existing columns

c) Add an integrity constraint d) To refine a column

Alter table tablename Add (new columnname datatype(size), new columnname datatype(size)...);

Alter table emp(add primary key(eno));

Alter table student(add constraint city_ chk check city in ('pune', 'mumbai'));

Solve the following

Set A :

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 02. 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.

Set B :

1. Consider the relation Employee created in assignment 03. Add a column join_date of type Date , with the constraint that joining date of any employee should always after 2002.

2. Consider the relation Department created in assignment 03. Add a column Dept_loc of type text , with the constraint that department location belongs to either Delhi or Mumbai or Bangalore.

| --- | --- | --- |
| **Signature of the Instructor** | **Remark** | **Date** |

Assignment: 05

Assignment to insert / update / delete records using tables created in previous assignments. (Use Si

(Use simple forms of insert / Update / Delete statements)

Insert Syntax:

Insert into tablename values(list of column values);

e.g.Insert into emp values(1,'joshi',4000,'pune);

Update Syntax:

Update tablename Set Columnname = Value where condition;

e.g. Update emp set sal = sal+1000 where eno =1;

Delete Syntax:

Delete from table_name where condition;

e.g. Delete from emp;

Delete from emp where eno = 1;

Solve the following.

Set A :

1. **Create the following tables (primary keys are underlined.).**

Property(pno,description,area) Owner(oname,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 clinet 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 client
c. change order date of client_no ’C004’ to 12/4/08
d. Delete all sale records having order date before 10th feb. 08
e. Delete the record of client “Joshi”

Set B :

Consider the tables created in assignments 1,2,3,4 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’, ’Natrajapt’,
’pune_nagarroad’ , ’Pune’,40014);

3. 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;

---------- ---------- ----------

Signature of the Instructor **Remark** **Date**

Assignment: 06

Assignment to query the tables using simple form of Select statement

Syntax:

select <attribute- list> from <relation- list> [where<condition>[group by <attribute list>

[having <condition>] order by<attribute list>]];

e.g.- Select * from emp;

- Select eno name from emp where sal > 4000 order by eno;
- Select sum(sal) from emp group by dno having sum(sal)> 100000;

The aggregate functions supported by as following

Name	Description	Usage	Example
Sum()	Total of the values of the attribute.	Sum(attribute-name)	Select sum(sal) from emp; Select dno, sum(sal) from emp group by dno;
Count()	Gives the count of members in the Group	Count(attribute); Count(*)	Select count(*) from emp; Select count(*) from emp where sal > 5000;
Max()	Gives the maximum value for an attribute, from a group of members	Max(attribute)	Select max(sal) from emp; Select dno, max(sal) from emp group by dno having count(*)>10;
Min()	Gives the minimum value for an attribute, from a group of Members	Min(attribute)	Select min(sal) from emp; Select dno, min(sal) from emp group by dno having min(sal) >100;
Avg()	Gives the average value for an attribute, from a group of Members	Avg(attribute)	Select avg(sal) from emp; Select dno, avg(sal) from emp group by dno having count(*) >10;

As part of the self activity in exercise you have created a table employee with attributes empno, name, address, salary and deptno. You have also inserted atleast 10 records into the same.

Set A:

Create the following relations person & area

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 business tasks and execute them.

1.List the names of all people living in 'Urban' area.
2.List details of all people whose names start with the alphabet 'C'
3. List the names of all people whose birthday falls in the month of March.
4.Give the count of people who are born on 12th June 1990
5.Give the count of people whose income is below < 5000.
6.Give the count of people whose income is between 10000 & 40000.
7.List the names of people with average income
8.List the sum of incomes of people living in 'Pune'
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 'Mumbai' and having income greater than 20000;
12.List the details pf 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 B:

Execute following select queries & write the business task performed by each query.

1. Select * from emp;
2. Select empno, name from emp;

3. Select distinct deptno from emp;
4. Select * from emp where deptno =101;
5. Select * from emp where address = 'Pune' and sal > 10000;
6. Select * from emp where address = 'Pune' and salary between 10000 and 30000 ;
7. Select * from emp where name like '_ _ _r%'
8. Select * from emp where name like 'P%' and 'S%';
9. Select * from emp where salary = Null;
10. Select * from emp order by eno;
11. Select * from emp order by deptno, eno desc;
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) > 4 order by deptno;
14. Select avg(salary) from emp;
15. Select max(salary),deptno from emp group by deptno having max(sal) >20000;
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');

---------- ---------- ----------

Signature of the Instructor **Remark** **Date**

Assignment: 07

Assignment to query table, using set operation (union, intersect)

SQL Set operations:

Name	Description	Syntax	Example
Union	Returns the union of two sets of values,eliminating duplicates.	<select query> Union <select query>	Select cname from depositor Union Select cname from borrower;
Union all	Returns the union of two sets of values, retaining all duplicates.	<select query> Union all <select query>	Select cname from depositor Union all Select cname from borrower;
Intersect	Returns the intersection oftwo sets of values, eliminating duplicates	<select query> intersect <select query>	Select cname from depositor intersect Select cname from borrower;
Intersect all	Returns the intersection of two sets of values, retaining duplicates	<select query> Intersect all <select query>	Select cname from depositor Intersect all Select cname from borrower;
Except	Returns the difference between two set of values, I.e returns all values from set1, not contained in set2.eliminates duplicates	<select query> except <select query>	Select cname from Depositor except Select cname from borrower;
Except all	Returns the difference between two set of values, i.e. returns all values from set1, not contained in set2 .Retains all Duplicates	<select query> Except all <select query>	Select cname from Depositor Except all Select cname from borrower;

The relations participating in the SQL operations union, intersect & except must be compatible the following two conditions must hold:

a) The relation r and must be of the same arity. That is, they must have the same number of attributes.

b) The domains of the i^{th} attribute of r and the i^{th} attribute of s must be the same , for all i.

Set A :

1.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 B:

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

One department 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 (empno int primary key, name varchar(20), address varchar(20), salary int,dno references department)

Teaching(empno int primary key, name varchar(20), address varchar(20), salary int,dno references department)

Department(dno int 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;

Signature of the Instructor

Remark

Date

Assignment: 08

Assignment to query tables using nested queries

A subquery 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' construc.
Set cardinality	The 'exists' construct returns thevalue 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)".

The complete Syntax of select statement containing connectivity or Comparison operators is as follows

```
select <attribute-list> from <relation-list>
    where <connectivity / comparison > { sub-query };
```

Set A :

1. Create the following relations :

Emp(eno,name,dno,salary) Project(pno,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 20000.
2. List the names of projects, controlled by department No , whose budget is greater than atleast one project controlled by department No 1 0 1 .

2. List the details of the projects with second maximum budget
3. List the details of the projects with third maximum budget.
4. List the names of employees, working on some projects that employee number is working.
5. List the names of employees who do not work on any project that employee number 204 works on.
6. List the names of employees who do not work on any project controlled by '102' department
7. List the names of projects along with the controlling department name, for those projects which has at least one employees working on it.
8. List the names of employees who is worked for more than 10 hrs on At least one project controlled by '104' dept. list the names of employees, who are males , and earning the maximum salary in their department.
9. List the names of employees who work in the same department as 'Clerk'.
10. List the names of employees who do not live in Pune or Mumbai.

Set B :

1. 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 ='Pune'));

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 where w.ename = e.name and c.company-name = w.company-name);

---------- ---------- ----------

Signature of the Instructor **Remark** **Date**

Assignment: 09

Assignment to query tables, using nested queries(use of exists, not exists)

SQL includes a feature for testing whether a sub query has any tuples in its result, using the following clauses:

Name	Description	Syntax	Example
Exists	The 'exists' construct returns the value true if the argument sub query is nonempty	select <attribute- list> from <relation- list> where <exists> { sub- query} ;	Select cname from borrower b where exists(select * from depositor where dname = b.cname);
Not exists	We can test for the non-existence of tuples in a sub query 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)"	select <attribute- list> from <relation- list> where <not exists>{ sub-query};	Select cname from borrower b where not exists(select * from depositor where dname = b.cname);

Set A:

1. Consider the table you have prepared as part of Assessment work set-A of exercise8,

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

   each query
2. List the names of employees who work in all the projects that "Mr. Pawar" works on.
3. List the names of employees who work on only some projects that "Mr. Joshi" works on
4. List the names of the departments that have at least one project under them. (Write using 'exists 'clause)
5. List the names of employees who do not work on "sales" project (write using 'not exists') clause
6. List the names of employees who work only on those projects that are controlled by their department.

7. List the names of employees who do not work on any projects that are controlled by their department.

Set B :

1. **Consider the table you have prepared as part of self activity of exercise 08 ,**

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

1. Select company-name from company c where not exists (select city from company where company-name = "_____" except (select city from company where company-name = c.company-name));

2. Select ename from employee e where exists (select manager-name from manages where manager name = e.ename group by manager-name having count(*) >3);

3. Select company-name from company c where not exists (select city from company where company-name = c.company-name except (select city from company where company-name = "_____"));

4. Select ename from employee e where exists (select city from employee where city= e.city and ename <> e.ename group by city having count(*) > 5)

5. Select company-name from company c where not exists (select company-name from company where city = c.city and company-name <> c.company-name)

---------- ---------- ----------

Signature of the Instructor **Remark** **Date**

Assignment: 10

Assignment related to small case studies (Each case study will involve creating tables with specified constraints, inserting records to it & writing queries for extracting records from these tables)

Steps in solving a case study:

- Read through the given case study carefully.
- Create the given relations in the database. The database thus created should be in 3NF. (no data duplication, appropriate handling of the relationships)
- Insert sufficient number of records in the relations / tables.
- Create a new file with all the select queries in it.
- To execute each query .

1. Consider the following case study:

A 4-wheeler rental company needs to develop a database to store the following information: The information about the cars , like the registration number, the chassis number, the type of the vehicle (car, jeep, SUV etc). the vehicles may have one or more luxurious features like AC, Stereo, tape, DVD player etc).

The company also needs to maintain the information about its drivers like driver license no, name, address, age etc.

A car is driven by different drivers on different days , a driver may drive different cars on different days . The company also needs information regarding the different places to which the car had been driven down, the names of drivers who have driven it to these places along with the name of customers who had booked the car to that place. The information of the different destinations to which the cars from this company can be driven down, also needs to be stored. Regarding customers, customers can book more than one car to a place. The customers are allowed to book multiple cars to different places, in a single booking transaction. The name, address, no of passengers travelling in the car, the destination, the rental cost etc needs to be stored.

The following constraints are to be defined for the vehicles, drivers, and destination places:

a) The vehicle make should be after the year 2000.

b) Only vehicles of maruti, Tata are used by the company

c) Drivers should be above 20 years of age
d) Drivers should be staying in “Pune” city
e) The destination places should be within 500km radius from Pune.

Design the relational database for the above company, so that the following queries can be answered:

1. List the names of drivers who have driven a car to “Mumbai”
2. List the name of customers who have booked a “SUV” to “Satara”
3. List the names of customers who have booked cars to Pune or Mumbai or Lonavla
4. List the details of cars that have never driven down to “Mumbai”
5. List the details of the place to which maximum number of customers have driven down.
6. List the details of the driver who have driven all the vehicles of the company.
7. List the names of the drivers who have driven at least two cars to “Mumbai
8. List the names of drivers who have also driven some vehicles to “Mumbai”
9. List the details of customers who have booked more than two vehicles to “Solapur”
10. List the names of customers who have booked maximum number of vehicles

2. Consider the following case study:

An insurance agent sells policies to clients. Each policy is of a particular type like vehicle insurance, life insurance, accident insurance etc, and there can be many policies of a particular type. Each policy will have many monthly premiums, and each premium is associated to only one policy. Assume appropriate attributes for agents, policy, premiums and policy-types.

The following constraints have to be defined on the relations

a. The policy types can be only accident, life and vehicle.
b. b. The agents can be only from Pune, Mumbai and Chennai.
c. c. The policy amount should be greater than 20000.
d. The policy-sale-date should be greater than the policy-intro-date.

Design the relational database for the above , so that the following queries can be answered:

1. List the names of agents living in ‘Nashik’
2. List the names of policy holders , who have bought policies from the agent ‘Mr.Joshi’
3. List the names of policyholders, who have bought more than two policies from ‘Mr.Joshi’
4. List the names of agents, who have sold policies to only customers who live in

their own City.

5.List the names of agents who have sold at least two policies.

6.List the names of cities, which have the maximum number of agents.

7.List the names of customers who have bought the maximum number of policies.

8.List the details of all premiums , paid or the policy number 2345.

9.Update all policy amount to 20000 for all policies bought by customers from Delhi city.

10.Delete all policies , bought from ‘Mr. Mane’

Signature of the Instructor

Remark

Date