DATABASE MANAGEMENT SYSTEM LABORATORY

COURSE CODE: BCSE2073

Lab Manual

for

BACHELOR OF

Engineering & Technology

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

GALGOTIAS UNIVERSITY, GREATER NOIDA

UTTAR PRADESH

Last revised on 04.08.2022

-Drafted by:
Dr. Nabanita Mahata

GALGOTIAS UNIVERSITY

Department of Computer Science & Engineering

Sr. No.	Title of Lab Experiments
1.	Implement Data Definition language Statements.
2.	Implement Data Manipulation Statements.
3.	Implement SELECT command with different clauses.
4.	Implement various type of Integrity Constraints on database.
5.	Implement SINGLE ROW functions (Character, Numeric, Date functions) and GROUP functions (avg, count, max, min, sum).
6.	Implement various type of SET OPERATORS (Union, Intersect, Minus) and JOINS.
7.	Implement the concept of grouping of Data and Subqueries.
8.	Implement the concept of Data Control Language (DCL), Transaction Control Language (TCL).
9.	Implement Simple and Complex View.
10.	Write a PL/SQL block to satisfy some conditions by accepting input from the user.
11.	Write a PL/SQL block for greatest of three numbers using IF AND ELSEIF
12.	Write a PL/SQL block for summation of odd numbers using for LOOP
13.	Write a PL/SQL Procedure for GCD Numbers
14.	Write a PL/SQL Procedure for cursor implementation
15.	Write a PL/SQL block to implementation of factorial using function
Value Added Experiments	
16.	Create a Database for Banking Sector and implement various queries on it.
17.	Create a Database for Customer Sale/purchase and implement various queries on it.

EXPERIMENT DETAILS

Experiment 1	
Title	Data Definition Language
Objective	Study of Data Definition language commands. - Create table, Alter Table, Drop Table, Rename Table.
Pre-requisite	Knowledge of Basic Database
Algorithm /Theory	The SQL DDL allows specification of not only a set of relations but also information about each relation, including- ● Schema for each relation ● The domain of values associated with each attribute. ● The integrity constraints. ● The set of indices to be maintained for each relation. ● The security and authorization information for each relation. The physical storage structure of each relation on disk.
Syntax	CREATE TABLE Emp1 (EID int, EName Char, Edept char, EDOB date,Salary int) Create Table Emp1(EID int, EName varchar(20), Edept varchar(10), EDOB Date, Salary int); CREATE TABLE TABLENAME (COLUMN_NAME1 DATA_TYPE1(SIZE1),……. COLUMN_NAMEN DATA_TYPEN(SIZEN)); ALTER TABLE ALTER TABLE *table_name* ADD *column_name datatype*; ALTER TABLE *table_name* MODIFY *column_name datatype*; ALTER TABLE *table_name* DROP COLUMN *column_name*; ALTER TABLE *table_name* RENAME COLUMN old_*column_name TO new_column_name*; DROP TABLE DROP TABLE *table_name*; RENAME TABLE RENAME old_*table_name TO new_table_name*; TRUNCATE TRUNCATE TABLE *table_name*;
Post Lab Assignme	

nt (If Any)	
Experiment 2	
Title	Data Manipulation Language Statements.
Objective	Study of Data Manipulation Statements.
Pre-requisite	Knowledge of ORACLE Queries
Algorithm /Theory	Data Manipulation Language (DML) statements are used for managing data in database. DML commands are not auto-committed. It means changes made by DML command are not permanent to database, it can be rolled back. DML statements are used for managing data within schema objects. Some examples: ○ SELECT - retrieve data from the a database ○ INSERT - insert data into a table ○ UPDATE - updates existing data within a table ○ DELETE - deletes all records from a table, the space for the records remain
Syntax	SELECT *column1*, *column2, ...*FROM *table_name*; INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...) VALUES (*value1*, *value2*, *value3*, ...); UPDATE *table_name* SET *column1* = *value1*, *column2* = *value2*, ... WHERE *condition*; DELETE FROM *table_name* WHERE *condition*; Delete from Emp where EID=2;

Post Lab Assignment (If Any)	
Experiment 3	
Title	SELECT Command
Objective	Study of SELECT command with different clauses.
Pre-requisite	Knowledge of • ORACLE
Algorithm /Theory	**SQL SELECT Statement** The most commonly used SQL command is SELECT statement. SQL SELECT statement is used to query or retrieve data from a table in the database. A query may retrieve information from specified columns or from all of the columns in the table. To create a simple SQL SELECT Statement, you must specify the column(s) name and the table name. The whole query is called SQL SELECT Statement.
Syntax	**Syntax of SQL SELECT Statement:** SELECT *column_list* FROM *table-name* [WHERE Clause] [GROUP BY clause] [HAVING clause] [ORDER BY clause];

Post Lab Assignment (If Any)	
Experiment 4	
Title	Keys
Objective	Study of various type of Integrity Constraints.
Pre-requisite	Knowledge of • ORACLE COMMANDS
Algorithm /Theory	SQL Constraints SQL constraints are used to specify rules for the data in a table. If there is any violation between the constraint and the data action, the action is aborted by the constraint. Constraints can be specified when the table is created (inside the CREATE TABLE statement) or after the table is created (inside the ALTER TABLE statement). In SQL, we have the following constraints: • **NOT NULL** - Indicates that a column cannot store NULL value • **UNIQUE** - Ensures that each row for a column must have a unique value • **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Ensures that a column (or combination of two or more columns) have a unique identity which helps to find a particular record in a table more easily and quickly • **FOREIGN KEY** - Ensure the referential integrity of the data in one table to match values in another table • **CHECK** - Ensures that the value in a column meets a specific condition • **DEFAULT** - Specifies a default value for a column SQL PRIMARY KEY Constraint The PRIMARY KEY constraint uniquely identifies each record in a database table. Primary keys must contain UNIQUE values. A primary key column cannot contain NULL values. Most tables should have a primary key, and each table can have only ONE primary key. SQL FOREIGN KEY Constraint A FOREIGN KEY in one table points to a PRIMARY KEY in another table.
Syntax	SQL CREATE TABLE + CONSTRAINT Syntax CREATE TABLE *table_name* (*column_name1 data_type*(*size*) *constraint_name*, *column_name2 data_type*(*size*) *constraint_name*, *column_name3 data_type*(*size*) *constraint_name*,

```
....
);
CREATE TABLE PersonsNotNull
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)

CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
PRIMARY KEY (P_Id)
)

CREATE TABLE Orders
(
O_Id int NOT NULL,
OrderNo int NOT NULL,
P_Id int,
PRIMARY KEY (O_Id),
FOREIGN KEY (P_Id) REFERENCES Persons(P_Id)
)
```

Post Lab Assignment (If Any)	
Experiment 5	
Title	SINGLE ROW functions and Group functions
Objective	Study of SINGLE ROW functions (Character, Numeric, Date functions) and GROUP functions (avg, count, max, min, sum).
Pre-requisite	Knowledge of ● ORACLE
Algorithm /Theory	Oracle SQL supplies a rich library of in-built functions which can be employed for various tasks. The essential capabilities of functions can be the case conversion of strings, in-string or substring operations, mathematical computations on numeric data, and date operations on date type values. SQL Functions optionally take arguments from the user and mandatorily return a value. Aggregate functions perform a variety of actions such as counting all the rows in a table, averaging a column's data, and summing numeric data. Aggregates can also search a table to find the highest "MAX" or lowest "MIN" values in a column. As with other types of queries, you can restrict, or filter out the rows these functions act on with the WHERE clause. For example, if a manager needs to know how many employees work in an organization, the aggregate function named COUNT(*) can be used to produce this information.The COUNT(*) function shown in the below SELECT statement counts all rows in a table.
Syntax	The SELECT query below demonstrates the use of NVL function. SELECT first_name, last_name, salary, NVL (commission_pct,0) FROM employees WHERE rownum < 5; FIRST_NAME LAST_NAME SALARY NVL(COMMISSION_PCT,0) -------------------- ------------------------- ---------- --------------------- Steven King 24000 0 Neena Kochhar 17000 0 Lex De Haan 17000 0 Alexander Hunold 9000 0 Some of the commonly used aggregate functions are as below - SUM([ALL | DISTINCT] expression) AVG([ALL | DISTINCT] expression)

	COUNT([ALL | DISTINCT] expression) COUNT(*) MAX(expression) MIN(expression)
Post Lab Assignment (If Any)	
Experiment 6(a)	
Title	SET Operators.
Objective	Study of various type of SET OPERATORS (Union, Intersect, Minus) and Various type of JOINS.
Pre-requisite	Knowledge of • Concept of SET Operators.
Algorithm /Theory	**Set Operation in SQL** SQL supports few Set operations to be performed on table data. These are used to get meaningful results from data, under different special conditions. **SQL JOIN** An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them. The most common type of join is: SQL INNER JOIN (simple join). An SQL INNER JOIN return all rows from multiple tables where the join condition is me. SQL INNER **JOIN Keyword** The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns in both table **SQL LEFT JOIN Keyword** The LEFT JOIN keyword returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match. **SQL RIGHT JOIN Keyword** The RIGHT JOIN keyword returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match. **SQL FULL OUTER JOIN Keyword** The FULL OUTER JOIN keyword returns all rows from the left table (table1) and from the right table (table2). FULL OUTER JOIN keyword combines the result of both LEFT and RIGHT joins.
Syntax	**select * from First** **UNION** **select * from second**

	SQL INNER JOIN Syntax **SELECT** ***column_name(s)*** **FROM** ***table1*** **INNER JOIN** ***table2*** **ON** ***table1.column_name=table2.column_name*****;**
Post Lab Assignment (If Any)	
Experiment 7	
Title	Subqueries
Objective	Study and implement the concept of sub queries.
Pre-requisite	Knowledge of • ORACAL COMMANDS
Algorithm /Theory	**Subqueries:- A** subquery is a form of an SQL statement that appears inside another SQL statement. It also termed as nested query. The statement containing a subquery called a parent statement. The rows returned bu the subquery are use by the following statement. It can be used by the following commands: 1. To insert records in the target table. 2. To create tables and insert records in this table. 3. To update records in the target table. 4. To create view. 5. To provide values for the condition in the WHERE, HAVING IN, SELECT, UPDATE, and DELETE statements. Exam:- Creating clientmaster table from oldclient_master, table Create table client_master AS SELECT * FROM oldclient_master;
Syntax	*Union Clause:* The user can put together multiple queries and combine their output using the union clause. The union clause merges the output of two or more queries into a single set of rows and column. The final output of union clause will be Output: = Records only in query one + records only in query two + A single set of records with is common in the both queries. Syntax:

SELECT columnname, columname
FROM tablename 1
UNION
SELECT columnname, columnname
From tablename2;

Intersect Clause: The use can put together multiple queries and their output using the interest clause. The final output of the interest clause will be :

Output =A single set of records which are common in both queries

Syntax:

SELECT columnname, columnname
FROM tablename 1
INTERSECT
SELECT columnname, columnname
FROM tablename 2;

MINUS CLAUSE:- The user can put together multiple queries and combine their output = records only in query one

Syntax:

SELECT columnname, columnname
FROM tablename ;
MINUS
SELECT columnname, columnname
FROM tablename ;

Post Lab Assignment (If Any)	
Experiment 8	
Title	Control languages
Objective	Study and implement the concept of Data Control Language (DCL), Transaction Control Language (TCL).
Pre-requisite	Knowledge of • ORACAL COMMANDS
Algorithm /Theory	**TCL command** Transaction Control Language(TCL) commands are used to manage transactions in database.These are used to manage the changes made by DML statements. It also allows statements to be grouped together into logical transactions. **Commit command** Commit command is used to permanently save any transaaction into database. Following is Commit command's syntax, **Rollback command** This command restores the database to last commited state. It is also use with savepoint command to jump to a savepoint in a transaction. Following is Rollback command's syntax, **Savepoint command** **savepoint** command is used to temporarily save a transaction so that you can rollback to that point whenever necessary. Following is savepoint command's syntax,
Syntax	***commit***; **rollback** to *savepoint-name*; **savepoint** *savepoint-name*;

Post Lab Assignment (If Any)	
Experiment 9	
Title	Views
Objective	Study of Simple and Complex View.
Pre-requisite	Knowledge of • ORACLE COMMANDS
Algorithm /Theory	CREATE VIEW Statement In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table. **Note:** A view always shows up-to-date data! The database engine recreates the data, using the view's SQL statement, every time a user queries a view.
Syntax	SQL CREATE VIEW Syntax CREATE VIEW view_name AS SELECT column_name(s) FROM table_name WHERE condition **Renaming the columns of a view:-** **Syntax:-** CREATE VIEW viewname AS SELECT newcolumnname…. FROM tablename WHERE columnname=expression_list; **Selecting a data set from a view-** **Syntax:-**

	SELECT columnname, columnname FROM viewname WHERE search condition; **Destroying a view-** **Syntax:-** DROP VIEW viewname;
Post Lab Assignment (If Any)	
Experiment 10	
Title	PL/SQL Program for Addition of Two numbers
Objective	PL/SQL Control Structure provides conditional tests, loops, flow control and branches that let to produce well-structured programs.
Pre-requisite	Knowledge of SQL
Algorithm /Theory	STEP 1: Start STEP 2: Initialize the necessary variables. STEP 3: Develop the set of statements with the essential operational parameters. STEP 4: Specify the Individual operation to be carried out. STEP 5: Execute the statements. STEP 6: Stop.

Syntax	SQL>set serveroutput on
	SQL>declare
	1 a number;
	2 b number;
	3 c number;
	4 begin
	5 a: =&a;
	6 b: =&b;
	7 c: =a+b;
	8 dbms_output.put_line ('sum of'||a||'and'||b||'is'||c);
	9 end;
	10 /
	INPUT
	Enter value for a: 23
	old 6: a:=&a;
	new 6: a:=23;

	Enter value for b: 12
	old 7: b:=&b;
	new 7: b:=12;
	OUTPUT sum of23and12is35
	PL/SQL procedure successfully completed.

Post Lab Assignment (If Any)	
Experiment 11	
Title	PL/SQL block for greatest of three numbers using IF AND ELSEIF
Objective	PL/SQL Control Structure provides conditional tests
Pre-requisite	Knowledge of SQL
Algorithm /Theory	STEP 1: Start STEP 2: Initialize the necessary variables. STEP 3: invoke the if else if condition. STEP 4: Execute the statements. STEP 5: Stop
Syntax	SQL>set server output on SQL> declare 2 a number; 3 b number; 4 c number; 5 begin 6 a:=&a; 7 b:=&b; 8 c:=&c; 9 if(a>b)and(a>c) then 10 dbms_output.put_line('A is maximum'); 11 else if(b>a)and(b>c)then 12 dbms_output.put_line('B is maximum'); 13 else 14 dbms_output.put_line('C is maximum'); 15 end if; 16 end;

	17 / **INPUT** Enter value for a: 21
	old 7: a:=&a;
	new 7: a:=21;
	Enter value for b: 12
	old 8: b:=&b;
	new 8: b:=12;
	Enter value for b: 45
	old 9: c:=&b;
	new 9: c:=45;
	OUTPUT C is maximum PL/SQL procedure successfully completed.

Post Lab Assignment (If Any)	
Experiment 12	
Title	PL/SQL block for summation of odd numbers using for LOOP
Objective	PL/SQL Control Structure provides conditional tests, loops, flow control and branches that let to produce well-structured programs.
Pre-requisite	Knowledge of SQL
Algorithm /Theory	STEP 1: Start STEP 2: Initialize the necessary variables. STEP 3: invoke the for loop condition. STEP 4: Execute the statements. STEP 5: Stop.
Syntax	SQL>set server output on SQL> declare 2 n number; 3 sum1 number default 0; 4 end value number; 5 begin 6 end value:=&end value; 7 n:=1; 8 for n in 1..endvalue 9 loop 10 if mod(n,2)=1 11 then 12 sum1:=sum1+n; 13 end if; 14 end loop;

	15 dbms_output.put_line('sum ='||sum1);
	16 end;
	17 /
	INPUT
	Enter value for end value: 4
	old 6: end value:=&end value;
	new 6: end value:=4;
	OUTPUT sum =4
	PL/SQL procedure successfully completed.

Post Lab Assignment (If Any)	
Experiment 13	
Title	PL/SQL Procedure for GCD Numbers
Objective	PL/SQL Control Structure provides conditional tests.
Pre-requisite	Knowledge of SQL

Algorithm /Theory	Create or replace procedure <procedure_name> (argument {in, out, in out} data type)
	{is, as} Variable declaration
	Begin
	Pl/SQL Subprogram body.
	Exception
	Exception PL/SQL Block.
	End;

Syntax	create or replace procedure pro is
	a number(3);
	b number(3);
	c number(3);
	d number(3);
	begin a:=&a;
	b:=&b;
	if(a>b) then c:=mod(a,b);
	if(c=0) then
	dbms_output.put_line('GCD is');
	dbms_output.put_line(b);
	else
	dbms_output.put_line('GCD is');

```
dbms_output.put_line(c);
 end if;
else d:=mod(b,a);
if(d=0) then
dbms_output.put_line('GCD is');
dbms_output.put_line(a);
else
dbms_output.put_line('GCD is');
dbms_output.put_line(d);
 end if;
end if;
end;
 /
 Enter value for a: 8
old 8: a:=&a;
new 8: a:=8;
 Enter value for b: 16
old 9: b:=&b;
 new 9: b:=16;
Procedure created.
 SQL> set serveroutput on;
SQL> execute pro;
GCD is 8
PL/SQL procedure successfully completed
```

Post Lab Assignment (If Any)	
Experiment 14	
Title	PL/SQL Procedure for cursor implementation.
Objective	To understand the concept of cursor.
Pre-requisite	Knowledge of SQL.

Algorithm /Theory Syntax	create table st13(regno number(4),name varchar2(20),mark1 number(3),mark2 number(3),mark3 numbe r(3),mark4 number(3),mark5 number(3));
	insert into st13 values(101,'raji',100,90,97,89,91);
	insert into a13 values(102,'kali');
	insert into a13 values(103,'jaya');
	select * from st13;
	REGNO NAME MARK1 MARK2 MARK3 MARK4 MARK5
	101 raji 100 90 97 89 91
	102 kali 99 77 69 81 99
	103 jaya 78 88 77 60 89
	SQL>set server output on
	declare
	ave number(5,2);
	tot number(3);
	cursor c_mark is select * from st13 where mark1>=40 and mark2>=40 and mark3>=40 and mark4>=40 and mark5>=40;
	begin
	dbms_output.put_line('regno name mark1 mark2 mark3 mark4 mark5 total average');
	dbms_output.put_line('----------');

	for student in c_mark
	loop
	tot:=st13.mark1+st13.mark2+st13.mark3+st13.mark4+st13.mark5;
	ave:=tot/5;
	dbms_output.put_line(st13.regno||rpad(st13.name,15)||rpad(st13.mark1,6)||rpad(st13.mark2,6)||rpad(st13.mark3,6)||rpad(st13.mark4,6)||rpad(st13.mark5,6)||rpad(tot,8)||rpad(ave,5)); end loop;end; /
	regno name mark1 mark2 mark3 mark4 mark5 total average
	--- 	
	101raji 100 90 97 89 91 467 93.4
	102kali 99 77 69 81 99 425 85
	103jaya 78 88 77 60 89 392 78.4
	PL/SQL procedure successfully completed.

Post Lab Assignment (If Any)	
Experiment 15	
Title	**FUNCTION TO FIND FACTORIAL**
Objective	To find factorial using function
Pre-requisite	Knowledge of SQL
Algorithm /Theory	Input An integer. Output Factorial of given number. Factorial(num) 1 if (num=0 or num=1) then. 2 fact = 1; 3 else. 4 for i 1 to n.
Syntax	SQL> create or replace function fact(n number) 2 return number is 3 i number(10); 4 f number:=1; 5 begin 6 for i in 1..N loop 7 f:=f*i; 8 end loop; 9 return f; 10 end;

	11 / Function created. SQL> select fact(2) from dual; FACT(2) --------- 2
Post Lab Assignment (If Any)	