



Y. Patil Pratishthan's

## **DR. D. Y. PATIL INSTITUTE OF ENGINEERING, MANAGEMENT & RESEARCH**

Approved by A.I.C.T.E, New Delhi , Maharashtra State Government, Affiliated to Savitribai Phule Pune University

Sector No. 29, PCNTDA , Nigidi Pradhikaran, Akurdi, Pune 411044. Phone: 020-27654470, Fax: 020-27656566  
Website : [www.dypiemr.ac.in](http://www.dypiemr.ac.in) Email : [principal.dypiemr@gmail.com](mailto:principal.dypiemr@gmail.com)

# **Department of Artificial Intelligence and Data Science LAB MANUAL Software Laboratory I (TE) Semester I**

**Prepared by:  
Mrs. Priyadarshani Doke  
Mrs. Archana Jadhav**



## Software Laboratory I

Course Code	Course Name	Teaching Scheme	Credits
317523	Software Laboratory I	4	2

### Course Objectives:

- To learn and apply various search strategies for AI
- To Formalize and implement constraints in search problems
- To develop basic Database manipulation skills
- To develop skills to handle NoSQL database
- To learn understand to develop application using SQL or NoSQL databases.

### Course Outcomes:

On completion of the course, learner will be able to–

- CO1: Implement SQL queries for given requirements, using different SQL concepts
- CO2: Implement NoSQL queries using MongoDB
- CO3: Design and develop application using database considering specific requirements
- CO4: Design a system using different informed search / uninformed search or heuristic approaches
- CO5: Apply basic principles of AI in solutions that require problem solving, inference, perception, knowledge representation, and learning.
- CO6: Design and develop an interactive AI application

**Operating System recommended:** 64-bit Open source Linux or its derivative

**Programming tools recommended:** MYSQL/Oracle, MongoDB, ERD plus, ER Win and use suitable programming language/Tool for implementation

## Table of Contents

Sr. No	Title of Experiment	CO Mapping	Page No

1	<p>SQL Queries:</p> <p>Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym, different constraints etc.</p> <p>Write at least 10 SQL queries on the suitable database application using SQL DML statements.</p>	CO1	7
2	<p>SQL Queries – all types of Join, Sub-Query and View:</p> <p>Write at least 10 SQL queries for suitable database application using SQL DML statements. Note: Instructor will design the queries which demonstrate the use of concepts like all types of Join, Sub-Query and View</p>	CO1	20
3	<p>MongoDB Queries:</p> <p>Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method,</p>	CO2	27
4	<p>Unnamed PL/SQL code block: Use of Control structure and Exception handling is mandatory.</p> <p>Suggested Problem statement:</p> <p>Consider Tables:</p> <ol style="list-style-type: none"> <li>1. Borrower (Roll_no, Name, Date_of_Issue, Name_of_Book, Status)</li> <li>2. Fine (Roll_no, Date, Amt)</li> </ol> <p>Accept Roll_no and Name_of_Book from user.  Check the number of days (from Date_of_Issue).  If days are between 15 to 30 then fine amount will be Rs 5 per day.  If no. of days &gt; 30, per day fine will be Rs 50 per day and for days less than 30, Rs. 5 per day.  After submitting the book, status will change from I to R.  If condition of fine is true, then details will be stored into fine table.  Also handles the exception by named exception handler or user define exception handler.</p> <p>OR</p> <p>MongoDB – Aggregation and Indexing: Design and Develop MongoDB Queries using aggregation and indexing with suitable example using MongoDB.</p> <p>MongoDB – Map-reduce operations: Implement Map-reduce operation with suitable example using MongoDB.</p>	CO1, CO2	36
6	<p>Cursors: (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor)</p> <p>Write a PL/SQL block of code using parameterized Cursor that will merge the data available in the newly created table N_Roll_Call with the data available in the table O_Roll_Call. If the data in the first table already exists in the second table then that data should be skipped.</p>	CO1	48
7	<p>Database Connectivity:</p> <p>Write a program to implement MySQL/Oracle database connectivity with any front end language to implement Database navigation operations (add, delete,</p>	CO1, CO3	56
<b>Group B</b>			
8	<p>Implement depth first search algorithm and Breadth First Search algorithm. Use an undirected graph and</p>	CO 4	61
9	<p>Implement A star (A*) Algorithm for any game search problem.</p>	CO 4	69

10	Implement Alpha-Beta Tree search for any game search problem.	CO 4	73
11	Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem.	CO 5	78
12	Implement Greedy search algorithm for any of the following application: Selection Sort Minimum Spanning Tree Single-Source Shortest Path Problem Job Scheduling Problem Prim's Minimal Spanning Tree Algorithm Kruskal's Minimal Spanning Tree Algorithm Dijkstra's Minimal Spanning Tree Algorithm	CO 5	85
13	Develop an elementary chatbot for any suitable customer interaction	CO 5, 6	94
14	Mini Project: Implement any one of the following Expert System Information management Hospitals and medical facilities Help desks management Employee performance evaluation Stock market trading Airline scheduling and cargo schedules	CO 5,6	99
<b>Group C</b>			
15	Develop an application with following details: 1. Follow the same problem statement decided in Assignment-1 of Group A. 2. Follow the Software Development Life cycle and other concepts learnt in Software Engineering Course throughout the implementation. 3. Develop application considering: Front End: Python/Java/PHP/Perl/Ruby/.NET/ or any other language Backend : MongoDB/ MySQL/ Oracle / or any standard SQL / NoSQL database 4. Test and validate application using Manual/Automation testing. 5. Student should develop application in group of 2-3 students and submit the Project Report which will consist of documentation related to different phases of Software Development Life Cycle: Title of the Project, Abstract, Introduction Software Requirement Specification (SRS) Conceptual Design using ER features, Relational Model in appropriate Normalize form Graphical User Interface, Source Code Testing document	CO1,CO3	107

# Group A



<b>Lab Assignment No.</b>	1
<b>Title</b>	SQL queries using Insert, Select, Update, delete with operators, functions, and set operator etc
<b>Roll No.</b>	
<b>Class</b>	TE
<b>Date of Completion</b>	
<b>Subject</b>	Software Laboratory I
<b>Assessment Marks</b>	
<b>Assessor's Sign</b>	

## ASSIGNMENT No: 01

**Title:** SQL queries using Insert, Select, Update, delete with operators, functions, and set operator etc

**Problem Statement:** Account(Acc\_no, branch\_name,balance)  
branch(branch\_name,branch\_city,assets) customer(cust\_name,cust\_street,cust\_city)  
Depositor(cust\_name,acc\_no) Loan(loan\_no,branch\_name,amount)

Borrower(cust\_name,loan\_no)

Solve following query:

Create above tables with appropriate constraints like primary key, foreign key, check constraints, not null etc.

Q1. Find the names of all branches in loan relation.

Q2. Find all loan numbers for loans made at Akurdi Branch with loan amount > 12000.

Q3. Find all customers who have a loan from bank. Find their names, loan\_no and loan amount.

Q4. List all customers in alphabetical order who have loan from Akurdi branch.

Q5. Find all customers who have an account or loan or both at bank.

Q6. Find all customers who have both account and loan at bank.

Q7. Find all customer who have account but no loan at the bank.

Q8. Find average account balance at Akurdi branch.

Q9. Find the average account balance at each branch

Q10. Find no. of depositors at each branch.

Q11. Find the branches where average account balance > 12000.

Q12. Find number of tuples in customer relation.

Q13. Calculate total loan amount given by bank.

Q14. Delete all loans with loan amount between 1300 and 1500.

Q15. Delete all tuples at every branch located in Nigdi.

Q.16. Create synonym for customer table as cust.

Q.17. Create sequence roll\_seq and use in student table for roll\_no column.

### **Objective:**

To Write SQL queries using Insert, Select, Update, delete with operators, functions, and set operator etc. Use of SQL objects such as Table, View, Index, Sequence, Synonym, different constraints etc.

### **Outcomes:**

Ability to write DDL and DML commands and their uses

### **Software Required:**

MYSQL/ORACLE



## Theory:

Introduction to SQL:

SQL stands for Structured Query Language

SQL lets you access and manipulate databases

SQL is an ANSI (American National Standards Institute) standard

Commands of SQL are grouped into four languages. 1>DDL

DDL is abbreviation of Data Definition Language. It is used to create and modify the structure of database objects in database.

Examples: CREATE, ALTER, DROP, RENAME, TRUNCATE statements

2>DML

DML is abbreviation of Data Manipulation Language. It is used to retrieve, store, modify, delete, insert and update data in database.

Examples: SELECT, UPDATE, INSERT, DELETE statements

3>DCL

DCL is abbreviation of Data Control Language. It is used to create roles, permissions, and referential integrity as well it is used to control access to database by securing it.

Examples: GRANT, REVOKE statements

4>TCL

TCL is abbreviation of Transactional Control Language. It is used to manage different transactions occurring within a database.

Examples: COMMIT, ROLLBACK statements

Data Definition Language (DDL)

1.Data definition Language (DDL) is used to create, rename, alter, modify, drop, replace, and delete tables, Indexes, Views, and comment on database objects; and establish a default database.

2.The DDL part of SQL permits database tables to be created or deleted. It also define indexes (keys), specify links between tables, and impose constraints between tables. The most important DDL statements in SQL are:

CREATE TABLE- Creates a new table

ALTER TABLE- Modifies a table

DROP TABLE- Deletes a table

TRUNCATE -Use to truncate (delete all rows) a table.

CREATE INDEX- Creates an index (search key)

DROP INDEX- Deletes an index

### 1. The CREATE TABLE Statement

The CREATE TABLE statement is used to create a table in a database.

Syntax

CREATE TABLE tablename

(attr1\_name attr1\_datatype(size) attr1\_constraint,  
attr2\_name attr2\_datatype(size) attr2\_constraint,...);

SQL Constraints

Constraints are used to limit the type of data that can go into a table.

Constraints can be specified when a table is created (with the CREATE TABLE statement) or after the table is created (with the ALTER TABLE statement).

We will focus on the following constraints:

NOT NULL

UNIQUE

PRIMARY KEY

FOREIGN KEY

CHECK

DEFAULT

Add constraint after table creation using alter table option

Syntax - Alter table add constraint constraint\_name constraint\_type(Attr\_name) Example -  
Alter table stud add constraint

prk1 primary key(rollno);

Drop constraint:

Syntax- Drop Constraint Constraint\_name; Example - Drop constraint prk1;

### 2. The Drop TABLE Statement Removes the table from the database Syntax

DROP TABLE table\_name;

### 3. The ALTER TABLE Statement

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

Syntax

To add a column in a table, use the following syntax:

```
ALTER TABLE table_name ADD column_name datatype;
```

To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

```
ALTER TABLE table_name DROP COLUMN column_name;
```

To change the data type of a column in a table, use the following syntax: ALTER TABLE table\_name

```
MODIFY COLUMN column_name datatype;
```

#### 4. The RENAME TABLE Statement

Rename the old table to new table;

Syntax

```
Rename old_tabname to new_tabname;
```

#### 5. The TRUNCATE TABLE Statement

The ALTER TABLE Statement is used to truncate (delete all rows) a table.

Syntax

To truncate a table, use following syntax : TRUNCATE TABLE table\_name;

#### 6. 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.

Syntax

```
CREATE VIEW view_name AS SELECT column_name(s) FROM table_name
```

```
WHERE condition;
```

#### 7. SQL Dropping a View

You can delete a view with the DROP VIEW command.

Syntax

```
DROP VIEW view_name;
```

#### 8 . Create Index Statement

1. Index in SQL is created on existing tables to retrieve the rows quickly. When there are thousands of records in a table, retrieving information will take a long time.
2. Therefore indexes are created on columns which are accessed frequently, so that the information can be retrieved quickly.
3. Indexes can be created on a single column or a group of columns. When an index is created, it first sorts the data and then it assigns a ROWID for each row.

Syntax

CREATE INDEX index\_name

ON table\_name (column\_name1,column\_name2...);

index\_name is the name of the INDEX.

table\_name is the name of the table to which the indexed column belongs.

column\_name1, column\_name2.. is the list of columns which make up the INDEX.

#### 9.Drop Index Statement Syntax

DROP INDEX index\_name;

#### 10. Create Synonym statement

1. Use the CREATE SYNONYM statement to create a synonym, which is an alternative name for a table, view, sequence, procedure, stored function, package, materialized view.
2. Synonyms provide both data independence and location transparency. Synonyms permit applications to function without modification regardless of which user owns the table or view and regardless of which database holds the table or view.
3. You can refer to synonyms in the following DML statements: SELECT, INSERT, UPDATE, DELETE

Syntax - Create synonym synonym-name for object-name;

Example-Create synonym synonym\_name for table\_name

Create synonym t for test

DML command

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.

#### 1) INSERT command

Insert command is used to insert data into a table. Following is its general syntax,

INSERT into table-name values(data1,data2,..)

Lets see an example,

Consider a table Student with following fields.

S\_id S\_Name age

INSERT into Student values(101,'Adam',15);

The above command will insert a record into Student table.

S\_id S\_Name age

101 Adam 15

## 2) UPDATE command

Update command is used to update a row of a table. Following is its general syntax,

UPDATE table-name set column-name = value where condition;

Lets see an example,

update Student set age=18 where s\_id=102;

## 3) Delete command

Delete command is used to delete data from a table. Delete command can also be used with condition to delete a particular row. Following is its general syntax,

DELETE from table-name;

Example to Delete all Records from a Table

DELETE from Student;

The above command will delete all the records from Student table.

Example to Delete a particular Record from a Table

Consider Student table

DELETE from Student where s\_id=103;

## SQL Functions

SQL provides many built-in functions to perform operations on data. These functions are useful while performing mathematical calculations, string concatenations, sub-strings etc. SQL functions are divided into two catagories,

- Aggregate Functions

- Scalar Functions

### Aggregate Functions

These functions return a single value after calculating from a group of values. Following are some frequently used Aggregate functions.

#### 1) AVG()

Average returns average value after calculating from values in a numeric column. Its general Syntax is,

SELECT AVG(column\_name) from table\_name e.g.

SELECT avg(salary) from Emp;

#### 2) COUNT()

Count returns the number of rows present in the table either based on some condition or without condition.

Its general Syntax is,

SELECT COUNT(column\_name) from table-name;

Example using COUNT()

Consider following Emp table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000

SQL query to count employees, satisfying specified condition is,

SELECT COUNT(name) from Emp where salary = 8000;

#### 3) FIRST()

First function returns first value of a selected column Syntax for FIRST function is,

SELECT FIRST(column\_name) from table-name

SQL query

SELECT FIRST(salary) from Emp;

#### 4) LAST()

LAST return the return last value from selected column Syntax of LAST function is,

SELECT LAST(column\_name) from table-name

SQL query will be,

```
SELECT LAST(salary) from emp;
```

#### 5) MAX()

MAX function returns maximum value from selected column of the table. Syntax of MAX function is,

```
SELECT MAX(column_name) from table-name
```

SQL query to find Maximum salary is,

```
SELECT MAX(salary) from emp;
```

#### 6) MIN()

MIN function returns minimum value from a selected column of the table. Syntax for MIN function is,

```
SELECT MIN(column_name) from table-name
```

SQL query to find minimum salary is,

```
SELECT MIN(salary) from emp;
```

#### 7) SUM()

SUM function returns total sum of a selected columns numeric values. Syntax for SUM is,

SELECT SUM(column\_name) from table-name SQL query to find sum of salaries will be,  
SELECT SUM(salary) from emp;

### Scalar Functions

Scalar functions return a single value from an input value. Following are some frequently used Scalar Functions.

#### 1) UCASE()

UCASE function is used to convert value of string column to Uppercase character. Syntax of UCASE,

```
SELECT UCASE(column_name) from table-name
```

#### Example of UCASE()

SQL query for using UCASE is,

```
SELECT UCASE(name) from emp;
```

#### 2) LCASE()

LCASE function is used to convert value of string column to Lowecase character.

Syntax for LCASE is:

SELECT LCASE(column\_name) from table-name

### 3) MID()

MID function is used to extract substrings from column values of string type in a table.

Syntax for MID function is:

SELECT MID(column\_name, start, length) from table-name

### 4) ROUND()

ROUND function is used to round a numeric field to number of nearest integer. It is used on Decimal point values. Syntax of Round function is,

SELECT ROUND(column\_name, decimals) from table-name

Operators:

AND and OR operators are used with Where clause to make more precise conditions for fetching data from database by combining more than one condition together.

#### 1) AND operator

AND operator is used to set multiple conditions with Where clause.

Example of AND

SELECT \* from Emp WHERE salary < 10000 AND age > 25

#### 2) OR operator

OR operator is also used to combine multiple conditions with Where clause. The only difference between AND and OR is their behaviour. When we use AND to combine two or more than two conditions, records satisfying all the condition will be in the result. But in case of OR, atleast one condition from the conditions specified must be satisfied by any record to be in the result.

Example of OR

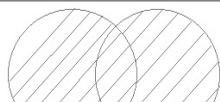
SELECT \* from Emp WHERE salary > 10000 OR age > 25

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.

#### 1) Union

UNION is used to combine the results of two or more Select statements. However it will eliminate duplicate rows from its result set. In case of union, number of columns and datatype must be same in both the tables.





Example of UNION

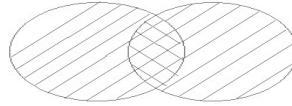
```
select * from First
```

UNION

```
select * from second
```

2) Union All

This operation is similar to Union. But it also shows the duplicate rows.



Union All query will be like,

```
select * from First
```

UNION ALL

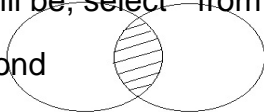
```
select * from second
```

3) Intersect

Intersect operation is used to combine two SELECT statements, but it only returns the records which are common from both SELECT statements. In case of Intersect the number of columns and datatype must be same. MySQL does not support INTERSECT operator.

Intersect query will be, ~~select \* from First~~ INTERSECT

```
select * from second
```

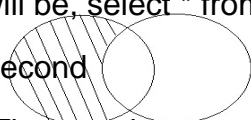


1) Minus

Minus operation combines result of two Select statements and return only those result which belongs to first set of result. MySQL does not support INTERSECT operator.

Minus query will be, ~~select \* from First~~ MINUS

```
select * from second
```



**Conclusion-** Thus, we have successfully studied DDL,DML commands and implemented SQL Queries.

<b>Lab Assignment No.</b>	2
<b>Title</b>	SQL queries using concepts like all types of Join, Sub-Query and View
<b>Roll No.</b>	
<b>Class</b>	TE
<b>Date of Completion</b>	
<b>Subject</b>	Software Laboratory I
<b>Assessment Marks</b>	
<b>Assessor's Sign</b>	

## ASSIGNMENT No: 02

**Title:** SQL queries using concepts like all types of Join, Sub-Query and View

### Problem Statement:

1. Create following Tables cust\_mstr(cust\_no,fname,lname)  
add\_dets(code\_no,add1,add2,state,city,pincode)

**Retrieve the address of customer Fname as 'xyz' and Lname as 'pqr'**

2. Create following Tables cust\_mstr(custno,fname,lname)  
acc\_fd\_cust\_dets(codeno,acc\_fd\_no) fd\_dets(fd\_sr\_no,amt)

**List the customer holding fixed deposit of amount more than 5000**

3. Create following Tables  
emp\_mstr(e\_mpno,f\_name,l\_name,m\_name,dept,desg,branch\_no) branch\_mstr(name,b\_no)

**List the employee details along with branch names to which they belong**

4. Create following Tables emp\_mstr(emp\_no,f\_name,l\_name,m\_name,dept)  
cntc\_dets(code\_no,cntc\_type,cntc\_data)

**List the employee details along with contact details using left outer join & right join**

5. Create following Tables cust\_mstr(cust\_no,fname,lname) add\_dets(code\_no,pincode)

**List the customer who do not have bank branches in their vicinity.**

6. a) Create View on borrower table by selecting any two columns and perform insert update delete operations  
b) Create view on borrower and depositor table by selecting any one column from each table perform insert update delete operations  
c) create updateable view on borrower table by selecting any two columns and perform insert update delete operations.

### Objective:

To write the SQL queries using concepts like all types of Join, Sub-Query and View

### Outcomes:

Able to write SQL queries using types of Join, Sub-Query and view

## Software Required:

MYSQL/ORACLE

## Theory:

### SQL JOIN

A JOIN clause is used to combine rows from two or more tables, based on a related column between them. Let's look at a selection from the "Orders" table:

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

Then, look at a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

Note that the "CustomerID" column in the "Orders" table refers to the "CustomerID" in the "Customers" table. The relationship between the two tables above is the "CustomerID" column. Then, we can create the following SQL statement (that contains an INNER JOIN), that selects records that have matching values in both tables:

### Example:

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate FROM Orders  
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

and it will produce something like this:

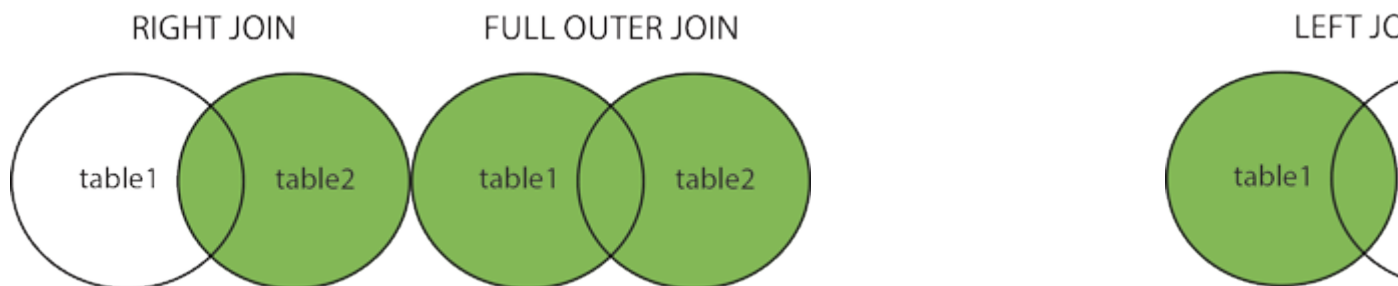
--	--	--

OrderID	CustomerName	OrderDate
10308	Ana Trujillo Emparedados y helados	9/18/1996
10365	Antonio Moreno Taquería	11/27/1996
10383	Around the Horn	12/16/1996
10355	Around the Horn	11/15/1996
10278	Berglunds snabbköp	8/12/1996

### Different Types of SQL JOINS:

Here are the different types of the JOINS in SQL:

- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table
- RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: Return all records when there is a match in either left or right table



SQL Views:

SQL 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.

CREATE VIEW Syntax:

CREATE VIEW view\_name AS SELECT column1, column2, ... FROM table\_name

WHERE condition;

SQL CREATE VIEW Examples:

The view "Product List" lists all active products (products that are not discontinued) from the "Products" table. The view is created with the following SQL:

```
CREATE VIEW ProductList AS SELECT ProductID, ProductName FROM Products
```

```
WHERE Discontinued = No;
```

Then, we can query the view as follows: `SELECT * FROM ProductList;`

SQL Updating a View:

You can update a view by using the following syntax:

```
SQL CREATE OR REPLACE VIEW Syntax CREATE OR REPLACE VIEW view_name AS  
SELECT column1, column2, ...
```

```
FROM table_name WHERE condition;
```

Now we want to add the "Category" column to the "Product List" view. We will update the view with the following SQL:

```
CREATE OR REPLACE VIEW Product List AS
```

```
SELECT ProductID, ProductName, Category FROM Products
```

```
WHERE Discontinued = No
```

Subqueries:

A subquery is a SQL query nested inside a larger query.

- A subquery may occur in :
  - - A SELECT clause
  - - A FROM clause
  - - A WHERE clause
- In MySQL subquery can be nested inside a SELECT, INSERT, UPDATE, DELETE, SET, or DO statement or inside another subquery.
- A subquery is usually added within the WHERE Clause of another SQL SELECT statement.
- You can use the comparison operators, such as >, <, or =. The comparison operator can also be a multiple-row operator, such as IN, ANY, SOME, or ALL.
- A subquery can be treated as an inner query, which is a SQL query placed as a part of

another query called as outer query.

- The inner query executes first before its parent query so that the results of the inner query can be passed to the outer query

### Subquery Syntax:

The subquery (inner query) executes once before the main query (outer query) executes.

The main query (outer query) use the subquery result.

Subquery syntax as specified by the SQL standard and supported in MySQL

```

SELECT select_list
FROM table
WHERE expr operator
      ( Select select_list
        FROM table );
DELETE FROM t1 WHERE s11 > ANY

```

(SELECT COUNT(\*) /\* no hint \*/ FROM t2

WHERE NOT EXISTS (SELECT \* FROM t3 WHERE ROW(5\*t2.s1,77)=

(SELECT 50,11\*s1 FROM t4 UNION SELECT 50,77 FROM (SELECT \* FROM t5) AS t5)));

A subquery can return a scalar (a single value), a single row, a single column, or a table (one or more rows of one or more columns). These are called scalar, column, row, and table subqueries.

### Subqueries: Guidelines

There are some guidelines to consider when using subqueries :

- A subquery must be enclosed in parentheses.
- Use single-row operators with single-row subqueries, and use multiple-row operators with multiple-row subqueries.
- If a subquery (inner query) returns a null value to the outer query, the outer query will not return any rows when using certain comparison operators in a WHERE clause.

### Types of Subqueries

- The Subquery as Scalar Operand
- Comparisons using Subqueries
- Subqueries with ALL, ANY, IN, or SOME
- Row Subqueries
- Subqueries with EXISTS or NOT EXISTS
- Correlated Subqueries

- Subqueries in the FROM Clause

**Conclusion:**

Thus, we have successfully studied Joins, Subqueries and views and implemented SQL Queries

<b>Lab Assignment No.</b>	3
<b>Title</b>	MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators etc)
<b>Roll No.</b>	
<b>Class</b>	TE
<b>Date of Completion</b>	
<b>Subject</b>	Software Laboratory I
<b>Assessment Marks</b>	
<b>Assessor's Sign</b>	



## ASSIGNMENT No: 03

**Title:** MongoDB Queries using CRUD operations. (Use CRUDOperations, SAVE method, logical operators etc)

### Problem Statement:

1. Create Database DYPIEMR
2. Create following Collections  
Teachers(Tname,dno,dname,experience,salary,date\_of\_joining )  
Students(Sname,roll\_no,class)
3. Find the information about all teachers
4. Find the information about all teachers of computer department
5. Find the information about all teachers of computer,IT,and e&TC department
6. Find the information about all teachers of computer,IT,and E&TC department having salary greater than or equal to 10000/-
7. Find the student information having roll\_no = 2 or Sname=xyz
8. Update the experience of teacher-praveen to 10years, if the entry is not available in database consider the entry as new entry.
9. Update the deparment of all the teachers working in IT department to COMP
10. find the teachers name and their experience from teachers collection
11. Using Save() method insert one entry in department collection
12. Using Save() method change the dept of teacher praveen to IT
13. Delete all the documents from teachers collection having IT dept.
14. display with pretty() method, the first 3 documents in teachers collection in ascending order

### Objective:

To Write the MongoDB Queries using CRUD operations. (Use CRUDOperations, SAVE method, logical operators etc)

**Outcome:**

Able to write MongoDB Queries using CRUD operations

**Software Required:**

MongoDB

**Theory:**

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

**Database**

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

**Collection**

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

**Document**

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

The following

table shows the relationship of RDBMS terminology with MongoDB

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)
<b>Database Server and Client</b>	
Mysqld/Oracle	mongod
mysql/sqlplus	mongo

CRUD is the basic operation of MongoDB, it stands CREATE, READ, UPDATE, DELETE.

## MongoDB 1. Create Collection

The createCollection() Method

MongoDB db.createCollection(name, options) is used to create collection. Basic syntax of createCollection() command is as follows: db.createCollection(name, options)

In the command, name is name of collection to be created. Options are a document and are used to specify configuration of collection.

Parameter	Type	Description
Options parameter is optional, so you need to specify only the name of the collection. Following is the list of options you can use:		
Name	String	Name of the collection to be created
Field	Type	Description
While inserting the document, MongoDB first checks size field of capped collection, then it checks max field.		
Example	Boolean	(Optional) If true, enables a capped collection. Capped collection is a fixed size collection that automatically overwrites its oldest entries when it reaches its maximum size. <b>If you specify true, you need to specify size parameter also.</b>
autoIndexID	Boolean	(Optional) If true, automatically create index on _id field. Default value is false.
size	number	(Optional) Specifies a maximum size in bytes for a capped collection. <b>If capped is true, then you need to specify this field also.</b>
max	number	(Optional) Specifies the maximum number of documents allowed in the capped collection.

You can check the created collection by using the command show collections.

```
>show collections mycollection
```

```
system.indexes
```

## 2. READ-The find() Method

To query data from MongoDB collection, you need to use MongoDB's find() method. Syntax

The basic syntax of find() method is as follows:

```
>db.COLLECTION_NAME.find()
```

find() method will display all the documents in a non-structured way. The pretty() Method

To display the results in a formatted way, you can use pretty() method. Syntax

```
>db.mycol.find().pretty() Example
>db.mycol.find().pretty()
{
  "_id": ObjectId(7df78ad8902c), "title": "MongoDB Overview",
  "description": "MongoDB is no sql database", "by": "tutorials point",
  "url": "http://www.tutorialspoint.com", "tags": ["mongodb", "database", "NoSQL"], "likes": "100"
}
>
```

Apart from find() method, there is findOne() method, that returns only one document.

### 3. UPDATE

MongoDB's update() and save() methods are used to update document into a collection.

The update() method updates the values in the existing document while the save() method replaces the existing document with the document passed in save() method.

#### MongoDB Update() Method

The update() method updates the values in the existing document. The basic syntax of update() method is as follows:

```
>db.COLLECTION_NAME.update(SELECTIOIN_CRITERIA, UPDATED_DATA)
```

#### Example

Consider the mycol collection has the following data.

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"} Following
example will set the new title 'New MongoDB Tutorial' of the documents whose title is
'MongoDB Overview'.
```

```
>db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}})
```

```
>db.mycol.find()
```

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"New MongoDB Tutorial"}
```

```
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
```

```
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

```
>
```

By default, MongoDB will update only a single document. To update multiple documents, you need to set a parameter 'multi' to true.

```
>db.mycol.update({'title':'MongoDB Overview'},  
{ $set: {'title':'New MongoDB Tutorial'}, {multi:true}})
```

### MongoDB Save() Method

The save() method replaces the existing document with the new document passed in the save() method.

The basic syntax of MongoDB save() method is

```
>db.COLLECTION_NAME.save({_id:ObjectId(),NEW_DATA}) Example
```

Following example will replace the document with the \_id '5983548781331adf45ec7'.

```
>db.mycol.save(  
{  
  "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point New Topic",  
  "by":"Tutorials Point"  
})  
  
>db.mycol.find()  
  
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"Tutorials Point New Topic", "by":"Tutorials Point"}  
  
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}  
  
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

### 4. DELETE-The remove() Method

MongoDB's remove() method is used to remove a document from the collection.

remove() method accepts two parameters. One is deletion criteria and second is justOne flag.

- deletion criteria: (Optional) deletion criteria according to documents will be removed.
- justOne: (Optional) if set to true or 1, then remove only one document. Basic syntax of remove() method is as follows:

```
>db.COLLECTION_NAME.remove(DELETION_CRITTERIA)
```

Example

Consider the mycol collection has the following data.

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
```

```
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
```

```
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

 Following example will remove all the documents whose title is 'MongoDB Overview'.

```
>db.mycol.remove({'title':'MongoDB Overview'})
```

```
>db.mycol.find()
```

```
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
```

```
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

## LOGICAL OPERATORS:

Syntax

### AND in MongoDB

In the find() method, if you pass multiple keys by separating them by ',' then MongoDB treats it as AND condition. Following is the basic syntax of AND

```
>db.mycol.find({key1:value1, key2:value2}).pretty() Example
```

Following example will show all the tutorials written by 'tutorials point' and whose title is 'MongoDB Overview'.

```
>db.mycol.find({"by":"tutorials point", "title": "MongoDB Overview"}).pretty()
```

```
{
```

```
  "_id": ObjectId(7df78ad8902c), "title": "MongoDB Overview",
```

```
  "description": "MongoDB is no sql database", "by": "tutorials point",
```

```
  "url": "http://www.tutorialspoint.com", "tags": ["mongodb", "database", "NoSQL"], "likes": "100"
```

For the above given example, equivalent where clause will be ' where by='tutorials point' AND title = 'MongoDB Overview' '. You can pass any number of key, value pairs in find clause.

### OR in MongoDB

Syntax : To query documents based on the OR condition, you need to use \$or keyword. Following is the basic syntax of OR

```
>db.mycol.find( { $or: [ {key1: value1}, {key2:value2} ] } ).pretty()
```

Example will show all the tutorials written by 'tutorials point' or whose title is 'MongoDB Overview'.

```
>db.mycol.find({$or:[{"by":"tutorials point"}, {"title": "MongoDB Overview"}]}).pretty()
{ "_id": ObjectId("7df78ad8902c"), "title": "MongoDB Overview",
  "description": "MongoDB is no sql database", "by": "tutorials point",
  "url": "http://www.tutorialspoint.com", "tags": ["mongodb", "database", "NoSQL"], "likes": "100" }
```

#### Using AND and OR Together Example

The following example will show the documents that have likes greater than 100 and whose title is either 'MongoDB Overview' or by is 'tutorials point'. Equivalent SQL where clause is 'where likes>10 AND (by = 'tutorials point' OR title = 'MongoDB Overview')'

```
>db.mycol.find({"likes": {$gt:10}, $or: [{"by": "tutorials point"}, {"title": "MongoDB Overview"}]}).pretty()
{
  "_id": ObjectId("7df78ad8902c"), "title": "MongoDB Overview",
  "description": "MongoDB is no sql database", "by": "tutorials point",
  "url": "http://www.tutorialspoint.com", "tags": ["mongodb", "database", "NoSQL"], "likes": "100" }
```

**Conclusion:** Thus we have studied MongoDB Queries using CRUD operations.

<b>Lab Assignment No.</b>	4
<b>Title</b>	A PL/SQL block of code (Use of Control structure and Exception handling)
<b>Roll No.</b>	
<b>Class</b>	TE
<b>Date of Completion</b>	
<b>Subject</b>	Software Laboratory I
<b>Assessment Marks</b>	
<b>Assessor's Sign</b>	

## ASSIGNMENT No: 04

**Title:** A PL/SQL block of code (Use of Control structure and Exception handling) Problem

### Problem Statement:

1. Consider table Stud(Roll, Att,Status)

Write a PL/SQL block for following requirement and handle the exceptions.

Roll no. of student will be entered by user. Attendance of roll no. entered by user will be checked in Stud table. If attendance is less than 75% then display the message "Term not granted" and set the status in stud table as "D". Otherwise display message "Term granted"



and set the status in stud table as “ND”

2. Write a PL/SQL block for following requirement using user defined exception handling.

The account\_master table records the current balance for an account, which is updated whenever, any deposits or withdrawals takes place. If the withdrawal attempted is more than the current balance held in the account. The user defined exception is raised, displaying an appropriate message. Write a PL/SQL block for above requirement using user defined exception handling.

3. 1. Borrower(Roll\_no, Name, Date\_of\_Issue, Name\_of\_Book, Status)

2. Fine(Roll\_no, Date, Amt)

- Accept Roll\_no & Name of Book from user.
- Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day.
- If no. of days>30, fine will be Rs 50 per day & for days less than 30, Rs. 5 per day.
- After submitting the book, status will change from I to R.
- If condition of fine is true, then details will be stored into fine table.

Also handles the exception by named exception handler or user define exception handler.

### **Objective:**

To write a PL/SQL block of code using Control structure and Exception handling.

### **Outcome:**

Able to understand and write code using Control structure and exception handling.

### **Software Required:**

MySQL/ORACLE

### **Theory:**

The PL/SQL architecture mainly consists of following 3 components:

1. PL/SQL block
2. PL/SQL Engine
3. Database Server

PL/SQL block:

- This is the component which has the actual PL/SQL code.
- This consists of different sections to divide the code logically (declarative section for declaring purpose, execution section for processing statements, exception handling section for handling errors)

- It also contains the SQL instruction that used to interact with the database server.
- All the PL/SQL units are treated as PL/SQL blocks, and this is the starting stage of the architecture which serves as the primary input.
- Following are the different type of PL/SQL units.
  - o Anonymous Block
  - o Function
  - o Library
  - o Procedure
  - o Package Body
  - o Package Specification
  - o Trigger
  - o Type
  - o Type Body

#### PL/SQL Engine

- PL/SQL engine is the component where the actual processing of the codes takes place.
- PL/SQL engine separates PL/SQL units and SQL part in the input (as shown in the image below).
- The separated PL/SQL units will be handled with the PL/SQL engine itself.
- The SQL part will be sent to database server where the actual interaction with database takes place.
- It can be installed in both database server and in the application server.

#### Database Server:

- This is the most important component of PL/SQL unit which stores the data.
- The PL/SQL engine uses the SQL from PL/SQL units to interact with the database server.
- It consists of SQL executor which actually parses the input SQL statements and execute the same.

#### Advantage of Using PL/SQL

1. Better performance, as sql is executed in bulk rather than a single statement
2. High Productivity
3. Tight integration with SQL

4. Full Portability
5. Tight Security
6. Support Object Oriented Programming concepts.

**Features of PL/SQL:-**

- 1) We can define and use variables and constants in PL/SQL.
- 2) PL/SQL provides control structures to control the flow of a program. The control structures supported by PL/SQL are if..Then, loop, for..loop and others.
- 3) We can do row by row processing of data in PL/SQL. PL/SQL supports row by row processing using the mechanism called cursor.
- 4) We can handle pre-defined and user-defined error situations. Errors are warnings and called as exceptions in PL/SQL.
- 5) We can write modular application by using sub programs.

**The structure of PL/SQL program:-**

The basic unit of code in any PL/SQL program is a block. All PL/SQL programs are composed of blocks. These blocks can be written sequentially.

**The structure of PL/SQL block:-****DECLARE**

Declaration section BEGIN

Executable section EXCEPTION

Exception handling section

END;

Where

- 1) Declaration section

PL/SQL variables, types, cursors, and local subprograms are defined here.

- 2) Executable section

Procedural and SQL statements are written here. This is the main section of the block. This section is required.

### 3) Exception handling section

Error handling code is written here

This section is optional whether it is defined within body or outside body of program.

### **Conditional statements and Loops used in PL/SQL**

Conditional statements check the validity of a condition and accordingly execute a set of statements. The conditional statements supported by PL/SQL is

#### 1) IF..THEN

#### 2) IF..THEN..ELSE

#### 3) IF..THEN..ELSIF

#### **1) IF..THEN**

Syntax1:-

If condition THEN Statement list END IF;

#### **2) IF..THEN..ELSE**

Syntax 2:-

IF condition THEN

Statement list

ELSE

Statements

#### **3) IF..THEN..ELSIF**

**END IF;**

Syntax 3:-

If condition THEN

Statement list ELSIF condition THEN

Statement list

ELSE

Statement list

END IF; END IF;

2) **CASE Expression** :CASE expression can also be used to control the branching logic within PL/SQL blocks. The general syntax is

CASE

WHEN <expression> THEN <statements>; WHEN <expression> THEN <statements>;

.

. ELSE

<statements>; END CASE;

Here expression in WHEN clause is evaluated sequentially. When result of expression is TRUE, then corresponding set of statements are executed and program flow goes to END CASE.

**ITERATIVE Constructs** : Iterative constructs are used to execute a set of statements respectively. The iterative constructs supported by PL/SQL are follows:

1) SIMPLE LOOP

2) WHILE LOOP

3) FOR LOOP

1) **The Simple LOOP** : It is the simplest iterative construct and has syntax like: LOOP

Statements

END LOOP;

The LOOP does not facilitate a checking for a condition and so it is an endless loop. To end the iterations, the EXIT statement can be used.

LOOP

<statement list>

```
IF condition THEN EXIT;  
END IF;  
END LOOP;
```

The statements here is executable statements, which will be executed repeatedly until the condition given if IF..THEN evaluates TRUE.

## 2) THE WHILE LOOP

The WHILE...LOOP is a condition driven construct i.e the condition is a part of the loop construct and

not to be checked separately. The loop is executed as long as the condition evaluates to TRUE. The syntax is:-

```
WHILE condition LOOP Statements  
END LOOP;
```

The condition is evaluated before each iteration of loop. If it evaluates to TRUE, sequence of statements are executed. If the condition is evaluated to FALSE or NULL, the loop is finished and the control resumes after the END LOOP statement.

3) **THE FOR LOOP** :The number of iterations for LOOP and WHILE LOOP is not known in advance. THE number of iterations depends on the loop condition. The FOR LOOP can be used to have a definite numbers of iterations.

The syntax is:-

Where

For loop counter IN [REVERSE] Low bound..High bound LOOP Statements;

End loop;

- loop counter –is the implicitly declared index variable as BINARY\_INTEGER.
- Low bound and high bound specify the number of iteration .
- Statements:-Are the contents of the loop

**EXCEPTIONS:-** Exceptions are errors or warnings in a PL/SQL program.PL/SQL implements

error handling using exceptions and exception handler.

Exceptions are the run time error that a PL/SQL program may encounter. There are two types of exceptions

- 1) Predefined exceptions
- 2) User defined exceptions

1) **Predefined exceptions:-** Predefined exceptions are the error condition that are defined by ORACLE. Predefined exceptions cannot be changed. Predefined exceptions correspond to common SQL errors. The predefined exceptions are raised automatically whenever a PL/SQL program violates an ORACLE rule.

2) **User defined Exceptions:-** A user defined exceptions is an error or a warning that is defined by the program. User defined exceptions can be define in the declaration section of PL/SQL block.

User defined exceptions are declared in the declarative section of a PL/SQL block. Exceptions have a type Exception and scope.

Syntax :

DECLARE

<Exception Name> EXCEPTION; BEGIN

....

RAISE <Exception Name>

...

EXCEPTION

WHEN <Exception name> THEN

<Action>

END;

### **Exception Handling**

A PL/SQL block may contain statements that specify exception handling routines. Each error or warning during the execution of a PL/SQL block raises an exception. One can distinguish between two types of exceptions:

- System dented exceptions
- User dented exceptions (which must be declared by the user in the declaration part of a

block where the exception is used/implemented)

System defined exceptions are always automatically raised whenever corresponding errors or warnings occur. User defined exceptions, in contrast, must be raised explicitly in a sequence of statements using `raise <exception name>`. After the keyword `exception` at the end of a block, user defined exception

handling routines are implemented. An implementation has the pattern `when <exception name> then <sequence of statements>;`

The most common errors that can occur during the execution of PL/SQL programs are handled by system defined exceptions.

Syntax:-

`<Exception_name>Exception;`

Handling Exceptions:- Exceptions handlers for all the exceptions are written in the exception handling section of a PL/SQL block.

Syntax:-

Exception

`When exception_name then Sequence_of_statements1;`

`When exception_name then Sequence_of_statements2;`

`When exception_name then Sequence_of_statements3;`

`End;`

Example:

Declare

`emp sal EMP.SAL%TYPE; emp no EMP.EMPNO%TYPE;`

`too_high_sal exception;`

`begin`

`select EMPNO, SAL into emp no, emp sal from EMP where ENAME = KING;`

`if emp sal 1.05 > 4000 then raise too high sal`

`else update EMP set SQL . . . end if ;`



exception

when NO DATA FOUND – – no tuple selected then rollback;

when too\_high\_sal then insert into high sal emps values(emp no); commit;

end;

After the keyword when a list of exception names connected with or can be specified. The lastwhen clause in the exception part may contain the exception name others. This introduces the default exception handling routine, for example, a rollback.

If a PL/SQL program is executed from the SQL\*Plus shell, exception handling routines may contain statements that display error or warning messages on the screen. For this, the procedure raise application error can be used. This procedure has two parameters <error number> and <message text>.

<error number> is a negative integer defined by the user and must range between -20000 and -20999.

<error message> is a string with a length up to 2048 characters.

The concatenation operator “||” can be used to concatenate single strings to one string. In order to display numeric variables, these variables must be converted to strings using the function to char. If the

procedure raise application error is called from a PL/SQL block, processing the PL/SQL block terminates and all database modifications are undone, that is, an implicit rollback is performed in addition to displaying the error message.

Example:

if emp sal 1.05 > 4000

then raise application error(-20010, Salary increase for employee with Id || to char (EMP no) || is too high);

E.g.

Declare

V\_maxno number (2):=20;

V\_curno number (2); E\_too\_many\_emp exception;

Begin

Select count (empno) into v\_curno from emp Where deptno=10;

```
If v_curno>25 then Raise e_too_many_Emp; End if;
```

```
Exception
```

```
when e_too_many_emp then
```

```
....
```

```
.....
```

```
end
```

**Conclusion:**Hence successfully learned to write PL/SQL code using control structure and exception handling

<b>Lab Assignment No.</b>	6
<b>Title</b>	PL/SQL code block using Cursors (All types: Implicit, Explicit,Cursor FOR Loop, Parameterized Cursor)
<b>Roll No.</b>	

<b>Class</b>	TE
<b>Date of Completion</b>	
<b>Subject</b>	Software Laboratory I
<b>Assessment Marks</b>	
<b>Assessor's Sign</b>	

## ASSIGNMENT No: 06

**Title:** A PL/SQL block of code (Use of Control structure and Exception handling) Problem

**Problem Statement:**

Implicit Cursor

1. The bank manager has decided to activate all those accounts which were previously marked as inactive for performing no transaction in last 365 days. Write a PL/SQ block (using implicit cursor) to update the status of account, display an approximate message based on the no. of rows affected by the update.

(Use of %FOUND, %NOTFOUND, %ROWCOUNT)

EXPLICIT CURSOR:

2. Organization has decided to increase the salary of employees by 10% of existing salary, who are having salary less than average salary of organization, Whenever such salary updates takes place, a record for the same is maintained in the increment\_salary table.

EMP (E\_no , Salary) increment\_salary(E\_no , Salary)

3. Write PL/SQL block using explicit cursor for following requirements:

College has decided to mark all those students detained (D) who are having attendance less than 75%. Whenever such update takes place, a record for the same is maintained in the D\_Stud table. create table stud21(roll number(4), att number(4), status varchar(1));

create table d\_stud(roll number(4), att number(4));

parameterized Cursor

4. Write a PL/SQL block of code using parameterized Cursor, that will merge the data available in the newly created table N\_RollCall with the data available in the table O\_RollCall.

If the data in the first table already exist in the second table then that data should be skipped.

parameterized Cursor

5. Write the PL/SQL block for following requirements using parameterized Cursor:

Consider table EMP(e\_no, d\_no, Salary), department wise average salary should be inserted into new table dept\_salary(d\_no, Avg\_salary)

EXPLICIT CURSOR: Cursor for loop

6. Write PL/SQL block using explicit cursor: Cursor FOR Loop for following requirements:

College has decided to mark all those students detained (D) who are having attendance less than 75%. Whenever such update takes place, a record for the same is maintained in the D\_Stud table. create table stud21(roll number(4), att number(4), status varchar(1));

create table d\_stud(roll number(4), att number(4));

### Objective:

To write a PL/SQL code block using Cursors (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor)

### Outcome:

Able to write PL/SQL code block using cursors

### Software Required:

MYSQL/ORACLE

### Theory:

#### • :- CURSOR:-

For the processing of any SQL statement, database needs to allocate memory. This memory is called context area. The context area is a part of PGA (Process global area) and is allocated on the oracle server.

A cursor is associated with this work area used by ORACLE, for multi row queries. A cursor is a handle or pointer to the context area. The cursor allows to process contents in the context area row by row.

There are two types of cursors.

1) Implicit cursor:-Implicit cursors are defined by ORACLE implicitly. ORACLE defines implicit cursor for every DML statements.

2) Explicit cursor:-These are user-defined cursors which are defined in the declaration section of the PL/SQL block. There are four steps in which the explicit cursor is processed.

1) Declaring a cursor

- 2) Opening a cursor
- 3) Fetching rows from an opened cursor
- 4) Closing cursor

General syntax for CURSOR:- DECLARE

Cursor cursor\_name IS select\_statement or query; BEGIN

Open cursor\_name;

Fetch cursor\_name into list\_of\_variables; Close cursor\_name;

END;

Where

- 1) Cursor\_name:-is the name of the cursor.
- 2) Select\_statement:-is the query that defines the set of rows to be processed by the cursor.
- 3) Open cursor\_name:-open the cursor that has been previously declared. When cursor is opened following things happen
  - i) The active set pointer is set to the first row.
  - ii) The value of the binding variables are examined.
- 4) Fetch statement is used to retrieve a row from the selected rows, one at a time, into PL/SQL variables.
- 5) Close cursor\_name:-When all of cursor rows have been retrieved, the cursor should be closed.

Explicit cursor attributes:-

Following are the cursor attributes

1. %FOUND: - This is Boolean attribute. It returns TRUE if the previous fetch returns a row and false if it doesn't.
2. %NOTFOUND:-If fetch returns a row it returns FALSE and TRUE if it doesn't. This as the exit condition for the fetch loop;  
is often used
3. %ISOPEN:-This attribute is used to determine whether or not the associated cursor is open. If so it returns TRUE otherwise FALSE.
4. %ROWCOUNT:-This numeric attribute returns a number of rows fetched by the cursor.

## Cursor Fetch Loops

### 1) Simple Loop Syntax:-

LOOP

Fetch cursorname into list of variables;

EXIT WHEN cursorname%NOTFOUND Sequence\_of\_statements;

END LOOP;

### 2) WHILE Loop Syntax:-

FETCH cursorname INTO list of variables; WHILE cursorname%FOUND LOOP

Sequence\_of\_statements;

FETCH cursorname INTO list of variables; END LOOP;

### 3) Cursor FOR Loop Syntax:

FETCH cursorname INTO list of variables; WHILE cursorname%FOUND LOOP

Sequence\_of\_statements;

FETCH cursorname INTO list of variables; END LOOP;

FOR variable\_name IN cursorname LOOP

-- an implicit fetch is done here.

-- cursorname%NOTFOUND is also implicitly checked.

-- process the fetch records. Sequence\_of\_statements;

END LOOP;

There are two important things to note about :-

i) Variable\_name is not declared in the DECLARE section. This variable is implicitly declared by the PL/SQL compiler.

ii) Type of this variable is cursorname%ROWTYPE.

## Implicit Cursors

PL/SQL issues an implicit cursor whenever you execute a SQL statement directly in your code, as long as that code does not employ an explicit cursor. It is called an "implicit" cursor because you, the developer, do not explicitly declare a cursor for the SQL statement.

If you use an implicit cursor, Oracle performs the open, fetches, and close for you automatically; these actions are outside of your programmatic control. You can, however, obtain information about the most recently executed SQL statement by examining the values in the implicit SQL cursor attributes.

PL/SQL employs an implicit cursor for each UPDATE, DELETE, or INSERT statement you execute in a program. You cannot, in other words, execute these statements within an explicit cursor, even if you want to. You have a choice between using an implicit or explicit cursor only when you execute a single-row SELECT statement (a SELECT that returns only one row).

In the following UPDATE statement, which gives everyone in the company a 10% raise, PL/SQL creates an implicit cursor to identify the set of rows in the table which would be affected by the update:

```
UPDATE employee
```

```
SET salary = salary * 1.1;
```

The following single-row query calculates and returns the total salary for a department. Once again, PL/SQL creates an implicit cursor for this statement:

```
SELECT SUM (salary) INTO department_total FROM employee
```

```
WHERE department_number = 10;
```

If you have a SELECT statement that returns more than one row, you must use an explicit cursor for that query and then process the rows returned one at a time. PL/SQL does not yet support any kind of array interface between a database table and a composite PL/SQL datatype such as a PL/SQL table.

### Drawbacks of Implicit Cursors

Even if your query returns only a single row, you might still decide to use an explicit cursor. The implicit cursor has the following drawbacks:

- It is less efficient than an explicit cursor
- It is more vulnerable to data errors
- It gives you less programmatic control

The following sections explore each of these limitations to the implicit cursor.

### Inefficiencies of implicit cursors

An explicit cursor is, at least theoretically, more efficient than an implicit cursor. An implicit

cursor executes as a SQL statement and Oracle's SQL is ANSI-standard. ANSI dictates that a single-row query must not only fetch the first record, but must also perform a second fetch to determine if too many rows will be returned by that query (such a situation will RAISE the TOO\_MANY\_ROWS PL/SQL exception). Thus, an implicit query always performs a minimum of two fetches, while an explicit cursor only needs to perform a single fetch.

This additional fetch is usually not noticeable, and you shouldn't be neurotic about using an implicit cursor for a single-row query (it takes less coding, so the temptation is always there). Look out for indiscriminate use of the implicit cursor in the parts of your application where that cursor will be executed repeatedly. A good example is the Post-Query trigger in the Oracle Forms.

Post-Query fires once for each record retrieved by the query (created from the base table block and the criteria entered by the user). If a query retrieves ten rows, then an additional ten fetches are needed with an implicit query. If you have 25 users on your system all performing a similar query, your server must process 250 additional (unnecessary) fetches against the database. So, while it might be easier to write an implicit query, there are some places in your code where you will want to make that extra effort and go with the explicit cursor.

#### Vulnerability to data errors

If an implicit SELECT statement returns more than one row, it raises the TOO\_MANY\_ROWS exception. When this happens, execution in the current block terminates and control is passed to the exception section. Unless you deliberately plan to handle this scenario, use of the implicit cursor is a declaration of faith. You are saying, "I trust that query to always return a single row!"

It may well be that today, with the current data, the query will only return a single row. If the nature of the data ever changes, however, you may find that the SELECT statement which formerly identified a single row now returns several. Your program will raise an exception. Perhaps this is what you will want. On the other hand, perhaps the presence of additional records is inconsequential and should be ignored.

With the implicit query, you cannot easily handle these different possibilities. With an explicit query, your program will be protected against changes in data and will continue to fetch rows without raising exceptions.

**Conclusion:**Hence successfully learn to implement cursor..



<b>Lab Assignment No.</b>	7
<b>Title</b>	Write a program to implement MySQL/Oracle database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)
<b>Roll No.</b>	
<b>Class</b>	TE
<b>Date of Completion</b>	
<b>Subject</b>	Software Laboratory I
<b>Assessment Marks</b>	
<b>Assessor's Sign</b>	

## ASSIGNMENT No: 07

**Title:** Implement MYSQL/Oracle database connectivity with PHP/ python/Java Implement Database navigation operations (add, delete, edit,) using ODBC/JDBC.

**Problem Statement:**

Write a program to implement MySQL/Oracle database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)

**Objective:** Ability to understand database connectivity using database and front end language

**Outcome:**

Able to understand database connectivity using database and front end language

**Software Required:**

Java,MYSQL



Output  
 sl2-pc5@sl2pc5-HP-Compaq-4000-Pro-SFF-PC:~\$ mysql -u root -p  
 Enter password:  
 Welcome to the MySQL monitor. Commands end with  
 ; or \g.  
 Your MySQL connection id is 42  
 Server version: 5.5.61-0ubuntu0.14.04.1 (Ubuntu) Copyright (c) 2000, 2018, Oracle and/or its  
 affiliates. All rights reserved.  
 Oracle is a registered trademark of Oracle  
 Corporation and/or its affiliates. Other names may be trademarks of their respective  
 owners.  
 Type 'help;' or '\h' for help. Type '\c' to clear the current input statement. mysql> create  
 database info;  
 Query OK, 1 row affected (0.03 sec)

```
mysql> use info;
Database changed
mysql> create table result (stud_RollNo int,stud_Name varchar(20),stud_Dept varchar(20));
Query OK, 0 rows affected (0.08 sec) mysql> select *from result;
```

	stud_RollNo	stud_Name	stud_Dept
1	abc	comp	

```
+ 1 row in set (0.00 sec)
//ADD DATA
mysql> select *from result;
```

	stud_RollNo	stud_Name	stud_Dept
1	abc	comp	
2	harsha	comp	
3	tej	comp	
4	rina	mech	

```
+4 rows in set (0.00 sec)
//DELETE DATA
mysql> select *from result;
```

	stud_RollNo	stud_Name	stud_Dept
2	harsha	comp	

```
|
3    | tej
| comp
|
4    | rina
| mech
+    +    +    + 3 rows in set (0.00 sec)
```

Conclusion: successfully learned to write code for database connectivity.

# Group B

<b>Lab Assignment No.</b>	8
<b>Title</b>	Implement depth first search algorithm and Breadth First Search algorithm. Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure.

<b>Roll No.</b>	
<b>Class</b>	TE
<b>Date of Completion</b>	
<b>Subject</b>	Software Laboratory I
<b>Assessment Marks</b>	
<b>Assessor's Sign</b>	

## ASSIGNMENT No: 08

**Title:** Implement depth first search algorithm and Breadth First Search algorithm.

**Problem Statement:** Implement depth first search algorithm and Breadth First Search algorithm. Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure.

### Objective:

Understand and implement BFS/DFS algorithm.

Analyze time complexity of the search algorithm.

### Outcomes:

Ability to choose an appropriate problem-solving method and knowledge representation technique.

### Software Required:

Python

### Theory:

Depth First Search:

Depth first Search or Depth first traversal is a recursive algorithm for searching all the vertices of a graph or tree data structure. Traversal means visiting all the nodes of a graph.

Depth First Search Algorithm:

A standard DFS implementation puts each vertex of the graph into one of two categories:

Visited

Not Visited

The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

The DFS algorithm works as follows:

- %3. Start by putting any one of the graph's vertices on top of a stack.
- %3. Take the top item of the stack and add it to the visited list.
- %3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of the stack.
- %3. Keep repeating steps 2 and 3 until the stack is empty.

Depth First Search Example:

Let's see how the Depth First Search algorithm works with an example. We use an undirected graph with 5 vertices.

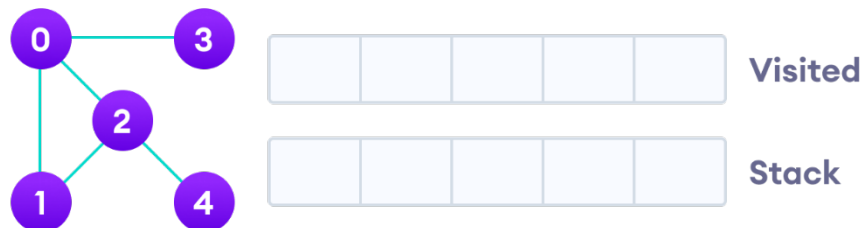


Figure 1: Undirected graph with 5 vertices

We start from vertex 0, the DFS algorithm starts by putting it in the Visited list and putting all its adjacent vertices in the stack.

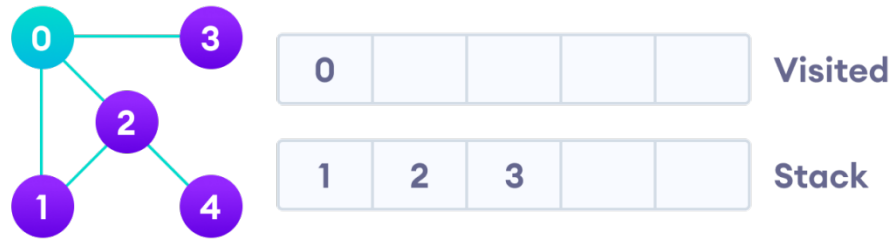


Figure 2: Visit the element and put it in the visited list

Next, we visit the element at the top of stack i.e. 1 and go to its adjacent nodes. Since 0 has already been visited, we visit 2 instead.

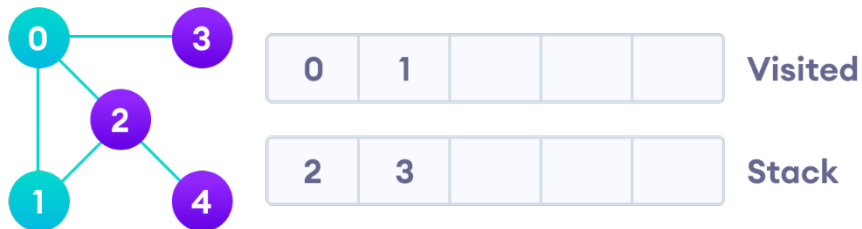


Figure 3: Visit the element at the top of stack

Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.

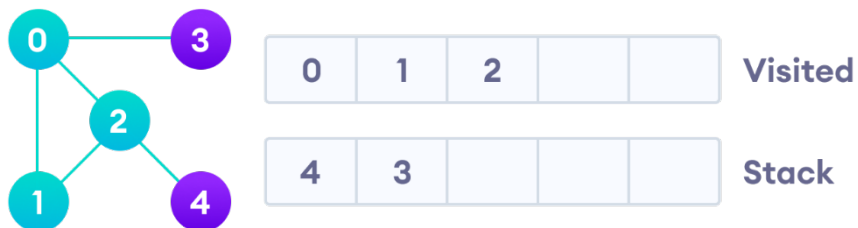


Figure 4: Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.



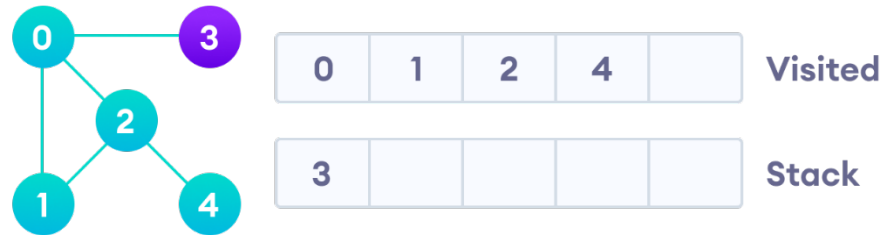


Figure 5: Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.

After we visit the last element 3, it doesn't have any unvisited adjacent nodes, so we have completed the Depth First Traversal of the graph.

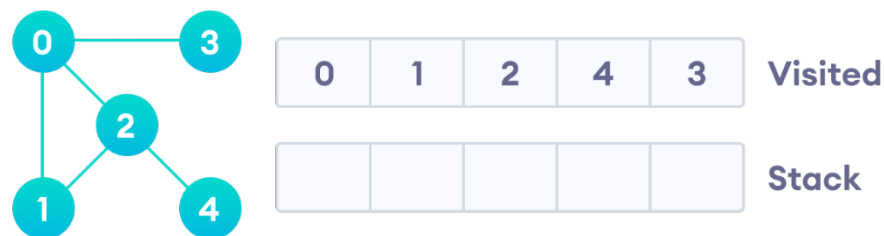


Figure 6: After we visit the last element 3, it doesn't have any unvisited adjacent nodes, so we have completed the Depth First Traversal of the graph.

DFS Pseudocode (recursive implementation):

The pseudocode for DFS is shown below. In the `init()` function, notice that we run the DFS function on every node. This is because the graph might have two different disconnected parts so to make sure that we cover every vertex, we can also run the DFS algorithm on every node.

```
DFS(G, u)
    u.visited = true
    for each v in G.Adj[u]
        if v.visited == false
            DFS(G,v)
```

```
init() {
```

```
For each u ∈ G
    u.visited = false
For each u ∈ G
    DFS(G, u)
}
```

#### Complexity of Depth First Search:

Time complexity of the DFS algorithm:  $O(V + E)$ ,  
where  $V$  is the number of nodes and  $E$  is the number of edges.  
Space complexity of the algorithm:  $O(V)$ .

#### Application of DFS Algorithm:

- For finding the path
- To test if the graph is bipartite
- For finding the strongly connected components of a graph
- For detecting cycles in a graph

#### Breadth First Search:

Traversal means visiting all the nodes of a graph. Breadth First Traversal or Breadth First Search is a recursive algorithm for searching all the vertices of a graph or tree data structure.

#### Breadth First Search Algorithm:

A standard BFS implementation puts each vertex of the graph into one of two categories:

- Visited
- Not Visited

The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

The BFS algorithm works as follows:

- %3. Start by putting any one of the graph's vertices at the back of a queue.
- %3. Take the front item of the queue and add it to the visited list.

%3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the back of the queue.

%3. Keep repeating steps 2 and 3 until the queue is empty.

#### Breadth First Search Example:

Let's see how the Breadth First Search algorithm works with an example. We use an undirected graph with 5 vertices.

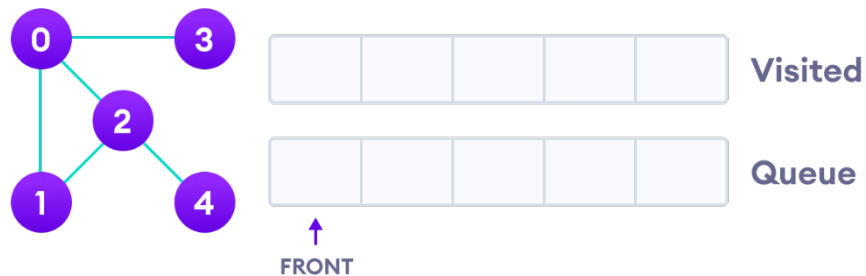


Figure 1: Undirected graph with 5 vertices

We start from vertex 0, the BFS algorithm starts by putting it in the Visited list and putting all its adjacent vertices in the queue.

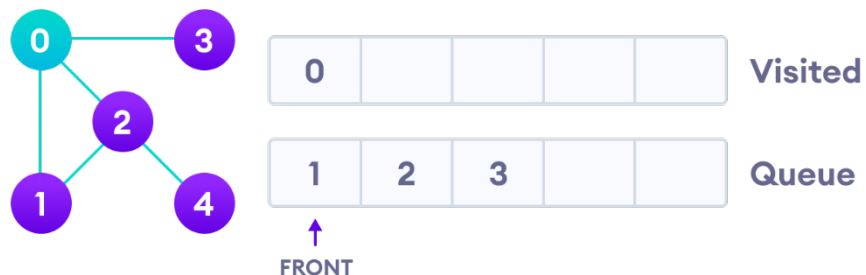


Figure 2: Visit start vertex and add its adjacent vertices to queue

Next, we visit the element at the front of queue i.e. 1 and go to its adjacent nodes. Since 0 has already been visited, we visit 2 instead.

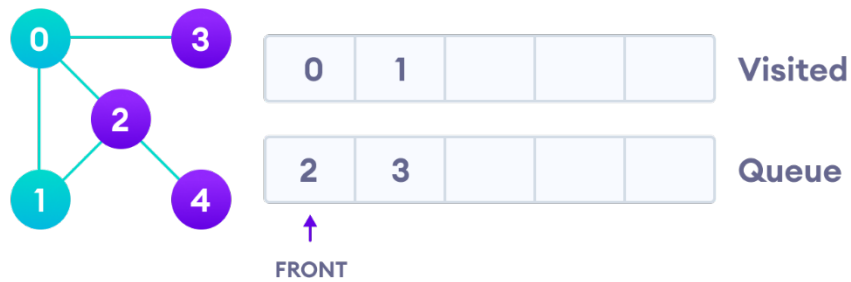


Figure 3: Visit the first neighbour of start node 0, which is 1

Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the back of the queue and visit 3, which is at the front of the queue.

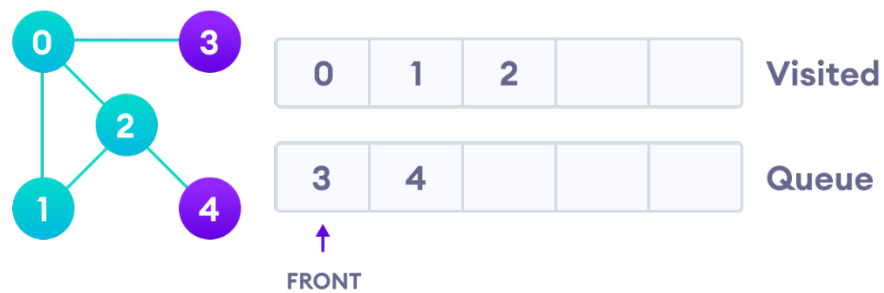


Figure 4: Visit 2 which was added to queue earlier to add its neighbors

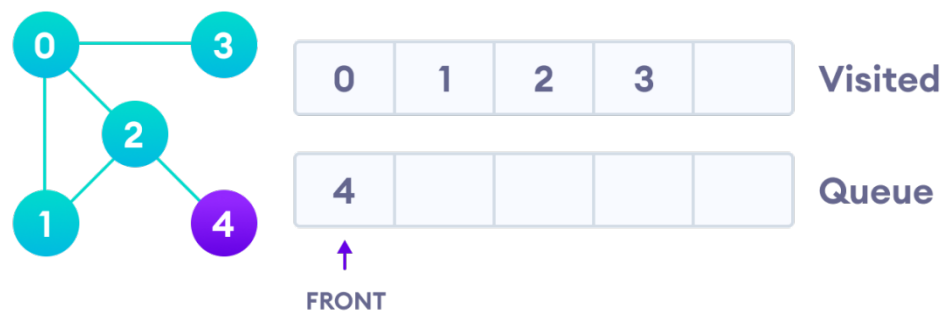


Figure 5: 4 remains in the queue

Only 4 remains in the queue since the only adjacent node of 3 i.e. 0 is already visited. We visit it.

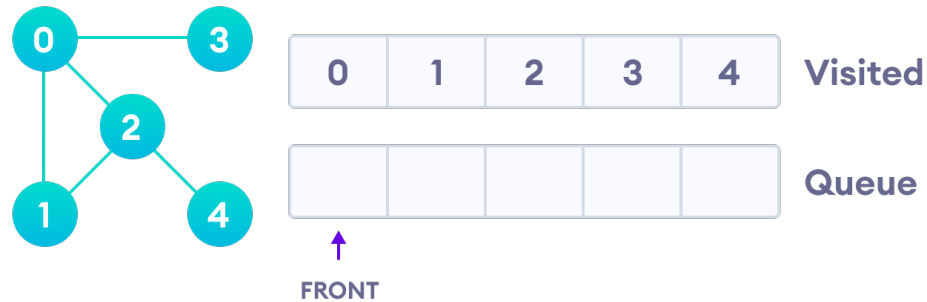


Figure 6: Visit last remaining item in the queue to check if it has unvisited neighbors

Since the queue is empty, we have completed the Breadth First Traversal of the graph.

BFS Pseudocode:

```

create a queue Q
mark v as visited and put v into Q
while Q is non-empty
    remove the head u of Q
    mark and enqueue all (unvisited) neighbours of u
  
```

Complexity of Breadth First Search:

Time complexity of the BFS algorithm:  $O(V + E)$ ,  
 where  $V$  is the number of nodes and  $E$  is the number of edges.  
 Space complexity of the BFS algorithm:  $O(V)$ .

Application of DFS Algorithm:

To build index by search index  
 For GPS navigation

Path finding algorithms

In Ford-Fulkerson algorithm to find maximum flow in a network

Cycle detection in an undirected graph

In minimum spanning tree

**Conclusion:** Implemented depth first search algorithm and Breadth First Search algorithm for an undirected graph using recursive algorithm for searching all the vertices of a graph or tree data structure.

<b>Lab Assignment No.</b>	9
<b>Title</b>	Implement A star (A*) Algorithm for any game search problem.
<b>Roll No.</b>	
<b>Class</b>	TE

<b>Date of Completion</b>	
<b>Subject</b>	Software Laboratory I
<b>Assessment Marks</b>	
<b>Assessor's Sign</b>	

## ASSIGNMENT No: 09

**Title:** Implement A star (A\*) Algorithm for any game search problem.

**Problem Statement:** Implement A star (A\*) Algorithm for any game search problem.

**Objective:**

Understand the working of informed search algorithm

Implement A\* search algorithm

**Outcome:**

Ability to choose an appropriate problem solving method and knowledge representation technique

**Software Required:**

Python

**Theory:**

An informed search strategy-one that uses problem-specific knowledge-can find solutions more efficiently.A key component of these algorithms is a heuristic function  $h(n)$   
 $h(n)$  = estimated cost of the cheapest path from node  $n$  to a goal node.

Admissible /heuristic never over estimated i.e.  $h(n)$  Actual cost. For example, Distance between two nodes(cities) $\Rightarrow$  straight line distance and for 8-puzzel problem- Admissible heuristic can be number of misplaced tiles  $h(n)= 8$ .

#### A\* Search technique:

It is informed search technique. It uses additional information beyond problem formulation and tree. Search is based on Evaluation function  $f(n)$ . Evaluation function is based on both heuristic function  $h(n)$  and  $g(n)$ .

$$f(n)=g(n) + h(n)$$

It uses two queues for its implementation: open, close Queue. Open queue is a priority queue which is arranged in ascending order of  $f(n)$

#### Algorithm:

1. Create a single member queue comprising of Root node
2. If FIRST member of queue is goal then goto step 5
3. If first member of queue is not goal then remove it from queue and add to close queue.
4. Consider its children if any, and add them to queue in ascending order of evaluation function  $f(n)$ .
5. If queue is not empty then goto step 2.
6. If queue is empty then goto step 6
7. Print 'success' and stop
8. Print 'failure' and stop.

#### Performance Comparison:

Completeness: yes

Optimality: yes

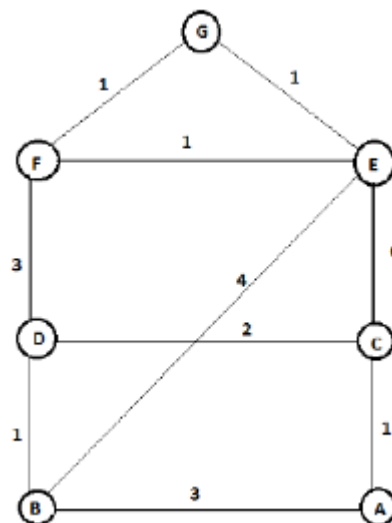
#### Limitation:

- It generate same node again and again
- Large Memory is required

#### Observation:

Although A\*generate many nodes it never uses those nodes for which  $f(n) > c^*$  where  $c^*$  is optimum cost.

Consider an example as below



CLOSE

OPEN/FRINGE



[ ]	[A]
[A]	[C,B]
[A,C]	[D,B,E,A]
[A,C,D]	[F,E,B,C,A]
[A,C,D,F]	[G,E,B,C,A,D]

SUCCESS

Node A:

$$f(B) = g(B) + h(B) = 3 + 5 = 8$$

$$f(C) = g(C) + h(C) = 1 + 6 = 7$$

Node C:

$$f(A) = g(A) + h(A) = 2 + 7 = 10$$

$$f(D) = g(D) + h(D) = 3 + 4 = 7$$

$$f(E) = g(E) + h(E) = 7 + 1 = 8$$

Node D:

$$f(F) = g(F) + h(F) = 6 + 1 = 7$$

$$f(C) = g(C) + h(C) = 5 + 6 = 11$$

$$f(B) = g(B) + h(B) = 4 + 5 = 9$$

Node F:

$$f(E) = g(E) + h(E) = 7 + 1 = 8$$

$$f(D) = g(D) + h(D) = 9 + 4 = 13$$

$$f(G) = g(G) + h(G) = 7 + 0 = 7$$

Final path: A C DF G

Total cost = 7

**Conclusion:** A good example of heuristic search is A\* search algorithm. The performance of such heuristic search algorithm depends upon the quality of heuristic function. A\* algorithm is executed and the path with minimum cost from source node to destination node is calculated.

<b>Lab Assignment No.</b>	10
<b>Title</b>	Implement Alpha-Beta Tree search for any game search problem.
<b>Roll No.</b>	
<b>Class</b>	TE
<b>Date of Completion</b>	
<b>Subject</b>	Software Laboratory I
<b>Assessment Marks</b>	
<b>Assessor's Sign</b>	

## ASSIGNMENT No: 10

**Title:** Implement Alpha-Beta Tree search for any game search problem.

**Problem Statement:** Implement Alpha-Beta Tree search for any game search problem.

**Objective:**

Study of optimization technique for minimax algorithm

Understand and implement Alpha-Beta Tree search for any game search problem.

Analyze time complexity of the search algorithm.

### Outcomes:

Ability to choose an appropriate problem-solving method and knowledge representation technique.

### Software Required:

Python

### Theory:

#### Alpha-Beta Pruning:

Alpha-Beta pruning is not actually a new algorithm, rather an optimization technique for minimax algorithm. It reduces the computation time by a huge factor. This allows us to search much faster and even go into deeper levels in the game tree. It cuts off branches in the game tree which need not be searched because there already exists a better move available. It is called Alpha-Beta pruning because it passes 2 extra parameters in the minimax function, namely alpha and beta.

Let's define the parameters alpha and beta.

Alpha is the best value that the maximizer currently can guarantee at that level or above.

Beta is the best value that the minimizer currently can guarantee at that level or above.

#### Pseudocode :

```
function minimax(node, depth, isMaximizingPlayer, alpha, beta):
```

```
    if node is a leaf node :
```

```
        return value of the node
```

```
    if isMaximizingPlayer :
```

```
        bestVal = -INFINITY
```

```
        for each child node :
```

```
            value = minimax(node, depth+1, false, alpha, beta)
```

```
            bestVal = max( bestVal, value)
```

```
            alpha = max( alpha, bestVal)
```

```
            if beta <= alpha:
```

```
                break
```

```
        return bestVal
```

```
    else :
```

```
        bestVal = +INFINITY
```

```
        for each child node :
```

```
            value = minimax(node, depth+1, true, alpha, beta)
```

```

    bestVal = min( bestVal, value)
    beta = min( beta, bestVal)
    if beta <= alpha:
        break
    return bestVal

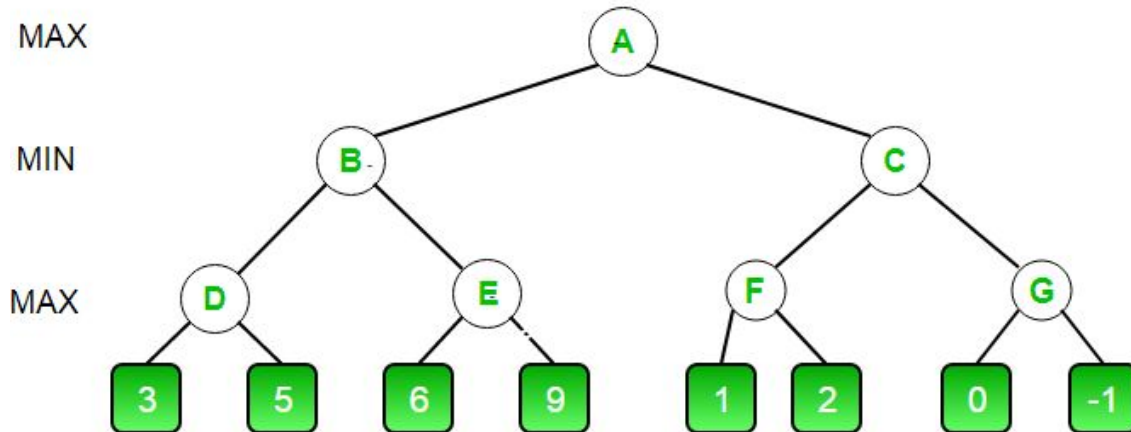
```

```

// Calling the function for the first time.
minimax(0, 0, true, -INFINITY, +INFINITY)

```

Example:



The initial call starts from A. The value of alpha here is -INFINITY and the value of beta is +INFINITY. These values are passed down to subsequent nodes in the tree.

At A the maximizer must choose max of B and C, so A calls B first

At B it the minimizer must choose min of D and E and hence calls D first.

At D, it looks at its left child which is a leaf node. This node returns a value of 3. Now the value of alpha at D is  $\max(-\text{INF}, 3)$  which is 3.

To decide whether its worth looking at its right node or not, it checks the condition  $\text{beta} \leq \text{alpha}$ . This is false since  $\text{beta} = +\text{INF}$  and  $\text{alpha} = 3$ . So it continues the search.

D now looks at its right child which returns a value of 5. At D,  $\text{alpha} = \max(3, 5)$  which is 5. Now the value of node D is 5

D returns a value of 5 to B. At B,  $\text{beta} = \min(+\text{INF}, 5)$  which is 5. The minimizer is now guaranteed a value of 5 or lesser. B now calls E to see if he can get a lower value than 5.

At E the values of alpha and beta is not -INF and +INF but instead -INF and 5 respectively, because the value of beta was changed at B and that is what B passed down to E

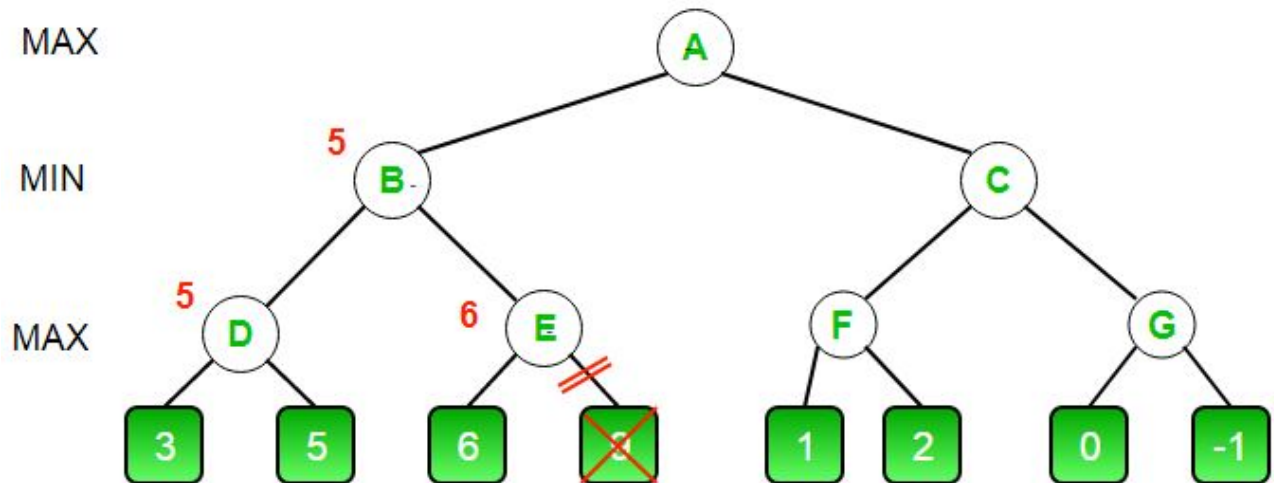
Now E looks at its left child which is 6. At E,  $\text{alpha} = \max(-\text{INF}, 6)$  which is 6. Here the condition becomes true. beta is 5 and alpha is 6. So  $\text{beta} \leq \text{alpha}$  is true. Hence it breaks and E returns 6 to B

Note how it did not matter what the value of E's right child is. It could have been +INF or -INF, it still wouldn't matter, We never even had to look at it because the minimizer was guaranteed a value of 5 or lesser. So as soon as the maximizer saw

the 6 he knew the minimizer would never come this way because he can get a 5 on the left side of B. This way we don't have to look at that 9 and hence saved computation time.

E returns a value of 6 to B. At B,  $\beta = \min(5, 6)$  which is 5. The value of node B is also 5

So far this is how our game tree looks. The 9 is crossed out because it was never computed.



B returns 5 to A. At A,  $\alpha = \max(-\infty, 5)$  which is 5. Now the maximizer is guaranteed a value of 5 or greater. A now calls C to see if it can get a higher value than 5.

At C,  $\alpha = 5$  and  $\beta = +\infty$ . C calls F

At F,  $\alpha = 5$  and  $\beta = +\infty$ . F looks at its left child which is a 1.  $\alpha = \max(5, 1)$  which is still 5.

F looks at its right child which is a 2. Hence the best value of this node is 2. Alpha still remains 5

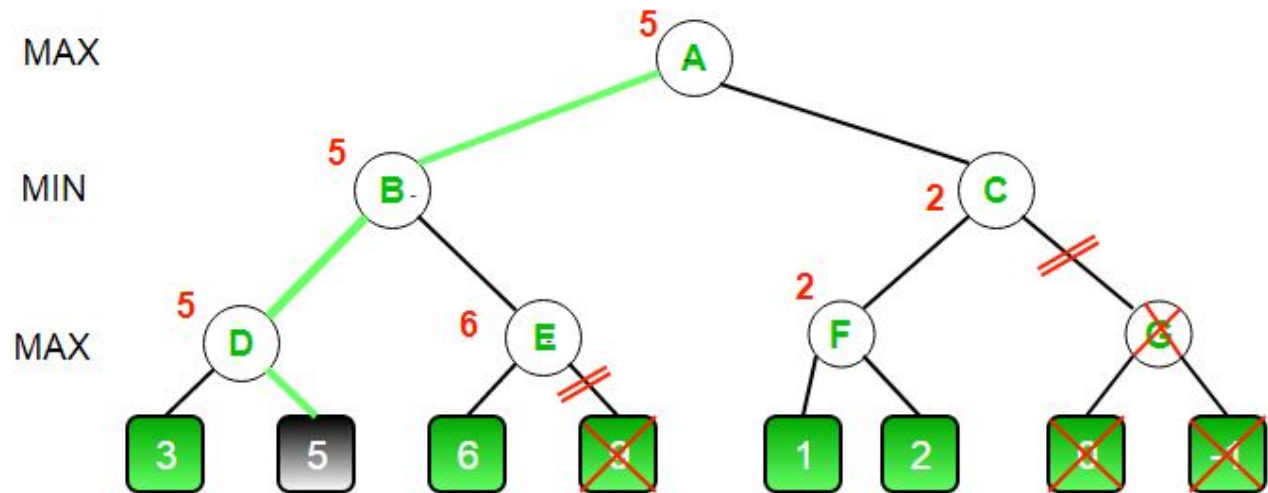
F returns a value of 2 to C. At C,  $\beta = \min(+\infty, 2)$ . The condition  $\beta \leq \alpha$  becomes true as  $\beta = 2$  and  $\alpha = 5$ . So it breaks and it does not even have to compute the entire sub-tree of G.

The intuition behind this break off is that, at C the minimizer was guaranteed a value of 2 or lesser. But the maximizer was already guaranteed a value of 5 if he choose B. So why would the maximizer ever choose C and get a value less than 2? Again you can see that it did not matter what those last 2 values were. We also saved a lot of computation by skipping a whole sub tree.

C now returns a value of 2 to A. Therefore the best value at A is  $\max(5, 2)$  which is a 5.

Hence the optimal value that the maximizer can get is 5

This is how our final game tree looks like. As you can see G has been crossed out as it was never computed.



**Conclusion:** Thus, we have implemented Alpha-Beta Tree search for any game search problem.

Lab Assignment No.	11
Title	Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem.
Roll No.	
Class	TE

<b>Date of Completion</b>	
<b>Subject</b>	Software Laboratory I
<b>Assessment Marks</b>	
<b>Assessor's Sign</b>	

## ASSIGNMENT No: 11

**Title:** Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem.

**Problem Statement:** Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem.

### Objective:

Understand and implement Constraint Satisfaction Problem using Branch and Bound for n-queens problem

Understand and implement Constraint Satisfaction Problem using Backtracking for n-queens problem

### Outcome:

Ability to choose an appropriate problem solving method and knowledge representation technique

### Software Required:

Python

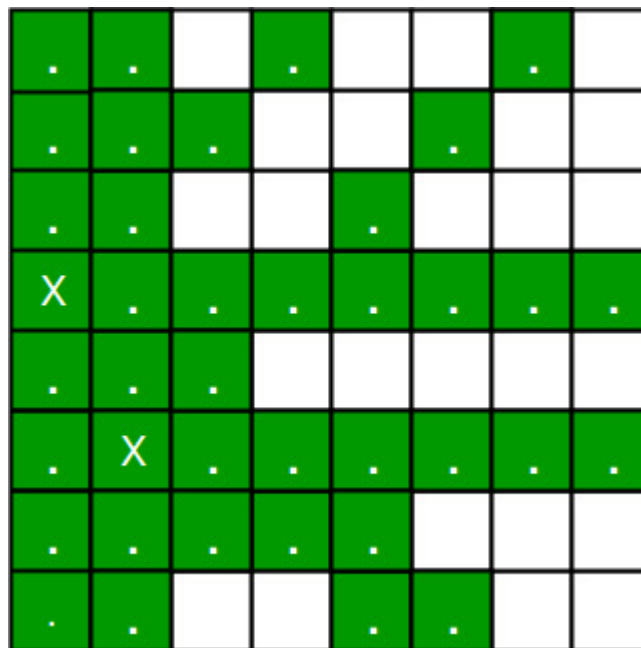
### Theory:

N Queen Problem using Branch and Bound:

The N queens puzzle is the problem of placing N chess queens on an NxN chessboard so that no two queens threaten each other. Thus, a solution requires that no two queens share the same row, column, or diagonal.

Backtracking Algorithm for N-Queen is already discussed here. In backtracking solution we backtrack when we hit a dead end. In Branch and Bound solution, after building a partial solution, we figure out that there is no point going any deeper as we are going to hit a dead end.

Let's begin by describing backtracking solution. "The idea is to place queens one by one in different columns, starting from the leftmost column. When we place a queen in a column, we check for clashes with already placed queens. In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution. If we do not find such a row due to clashes, then we backtrack and return false."





•	•		•			•	
•	•	•			•		
•	•			•			
X	•	•	•	•	•	•	•
•	•	•					
•	X	•	•	•	•	•	•
•	•	•	•	•			
•	•			•	•		

1. For the 1st Queen, there are total 8 possibilities as we can place 1st Queen in any row of first column. Let's place Queen 1 on row 3.
2. After placing 1st Queen, there are 7 possibilities left for the 2nd Queen. But wait, we don't really have 7 possibilities. We cannot place Queen 2 on rows 2, 3 or 4 as those cells are under attack from Queen 1. So, Queen 2 has only  $8 - 3 = 5$  valid positions left.
3. After picking a position for Queen 2, Queen 3 has even fewer options as most of the cells in its column are under attack from the first 2 Queens.

We need to figure out an efficient way of keeping track of which cells are under attack. In previous solution we kept an 8-by-8 Boolean matrix and update it each time we placed a queen, but that required linear time to update as we need to check for safe cells.

Basically, we have to ensure 4 things:

1. No two queens share a column.
2. No two queens share a row.
3. No two queens share a top-right to left-bottom diagonal.
4. No two queens share a top-left to bottom-right diagonal.

Number 1 is automatic because of the way we store the solution. For number 2, 3 and 4, we can perform updates in  $O(1)$  time. The idea is to keep three Boolean arrays that tell us which rows and which diagonals are occupied.

Lets do some pre-processing first. Let's create two  $N \times N$  matrix one for / diagonal and other one for \ diagonal. Let's call them slashCode and backslashCode respectively. The trick is to fill them in such a way that two queens sharing a same /-diagonal will have the same value in matrix slashCode, and if they share same

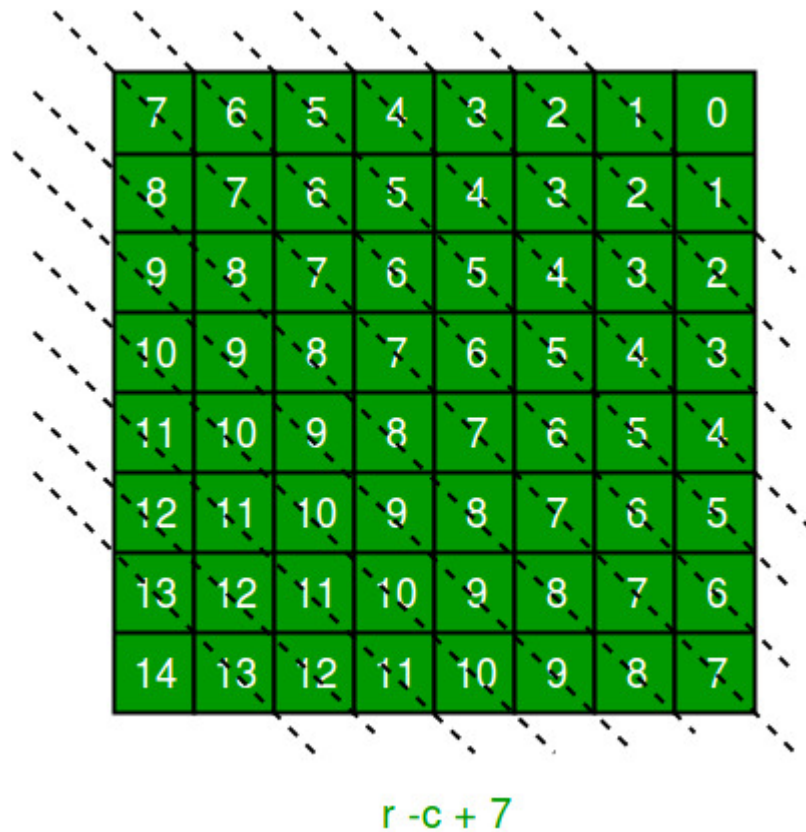
\-diagonal, they will have the same value in backslashCode matrix.

For an N x N matrix, fill slashCode and backslashCode matrix using below formula –

$$\text{slashCode}[\text{row}][\text{col}] = \text{row} + \text{col}$$

$$\text{backslashCode}[\text{row}][\text{col}] = \text{row} - \text{col} + (\text{N}-1)$$

Using above formula will result in below matrices



7	6	5	4	3	2	1	0
8	7	6	5	4	3	2	1
9	8	7	6	5	4	3	2
10	9	8	7	6	5	4	3
11	10	9	8	7	6	5	4
12	11	10	9	8	7	6	5
13	12	11	10	9	8	7	6
14	13	12	11	10	9	8	7

$r - c + 7$

0	1	2	3	4	5	6	7
1	2	3	4	5	6	7	8
2	3	4	5	6	7	8	9
3	4	5	6	7	8	9	10
4	5	6	7	8	9	10	11
5	6	7	8	9	10	11	12
6	7	8	9	10	11	12	13
7	8	9	10	11	12	13	14

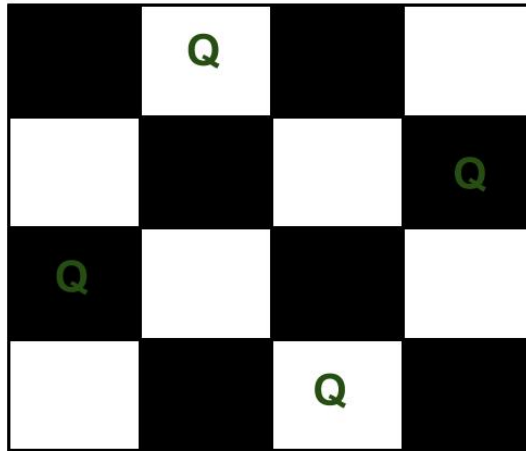
$r + c$

The 'N - 1' in the backslash code is there to ensure that the codes are never negative because we will be using the codes as indices in an array.

Now before we place queen  $i$  on row  $j$ , we first check whether row  $j$  is used (use an array to store row info). Then we check whether slash code ( $j + i$ ) or backslash code ( $j - i + 7$ ) are used (keep two arrays that will tell us which diagonals are occupied). If yes, then we have to try a different location for queen  $i$ . If not, then we mark the row and the two diagonals as used and recurse on queen  $i + 1$ . After the recursive call returns and before we try another position for queen  $i$ , we need to reset the row, slash code and backslash code as unused again, like in the code from the previous notes.

### N Queen Problem using Backtracking:

The N Queen is the problem of placing N chess queens on an  $N \times N$  chessboard so that no two queens attack each other. For example, the following is a solution for the 4 Queen problem.



The expected output is a binary matrix that has 1s for the blocks where queens are placed. For example, the following is the output matrix for the above 4 queen solution.

{ 0, 1, 0, 0}

{ 0, 0, 0, 1}

{ 1, 0, 0, 0}

{ 0, 0, 1, 0}

#### Backtracking Algorithm:

The idea is to place queens one by one in different columns, starting from the leftmost column. When we place a queen in a column, we check for clashes with already placed queens. In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution. If we do not find such a row due to clashes, then we backtrack and return false.

1. Start in the leftmost column
2. If all queens are placed

return true

3. Try all rows in the current column.

Do following for every tried row.

- a) If the queen can be placed safely in this row then mark this [row, column] as part of the solution and recursively check if placing queen here leads to a solution.
  - b) If placing the queen in [row, column] leads to a solution then return true.
  - c) If placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to step (a) to try other rows.
4. If all rows have been tried and nothing worked, return false to trigger backtracking.

**Conclusion:** Thus we have Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem.

Lab Assignment No.	12
-----------------------	----

<b>Title</b>	Implement Greedy search algorithm for any of the following application: Selection Sort Minimum Spanning Tree Single-Source Shortest Path Problem Job Scheduling Problem Prim's Minimal Spanning Tree Algorithm Kruskal's Minimal Spanning Tree Algorithm Dijkstra's Minimal Spanning Tree Algorithm
<b>Roll No.</b>	
<b>Class</b>	TE
<b>Date of Completion</b>	
<b>Subject</b>	Software Laboratory I
<b>Assessment Marks</b>	
<b>Assessor's Sign</b>	

## ASSIGNMENT No: 12

**Title:** Implement Greedy search algorithm for any of the following application:

Selection Sort

Minimum Spanning Tree

Single-Source Shortest Path Problem

Job Scheduling Problem

Prim's Minimal Spanning Tree Algorithm

Kruskal's Minimal Spanning Tree Algorithm

Dijkstra's Minimal Spanning Tree Algorithm

**Problem Statement:** Implement Greedy search algorithm for any of the following application:

Prim's Minimal Spanning Tree Algorithm

Kruskal's Minimal Spanning Tree Algorithm

**Objective:**

Prim's Minimal Spanning Tree Algorithm

Kruskal's Minimal Spanning Tree Algorithm

**Outcome:**

Ability to choose an appropriate problem solving method and knowledge representation technique

**Software Required:**

Python

**Theory:**

Kruskal's Minimum Spanning Tree Algorithm:

What is a Spanning Tree?

A Spanning tree is a subset to a connected graph  $G$ , where all the edges are connected, i.e, we can traverse to any edge from a particular edge with or without intermediates. Also, a spanning tree must not have any cycle in it. Thus we can say that if there are  $n$  vertices in a connected graph then the no. of edges that a spanning tree may have is  $n-1$ .

What is Minimum Spanning Tree?

Given a connected and undirected graph, a spanning tree of that graph is a subgraph that is a tree and connects all the vertices together. A single graph can have many different spanning trees. A minimum spanning tree (MST) or minimum weight spanning tree for a weighted, connected, undirected graph is a spanning tree with a weight less than or equal to the weight of every other spanning tree. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree.

How many edges does a minimum spanning tree has?

A minimum spanning tree has  $(V - 1)$  edges where  $V$  is the number of vertices in the given graph.

What are the applications of the Minimum Spanning Tree?

See this for applications of MST.

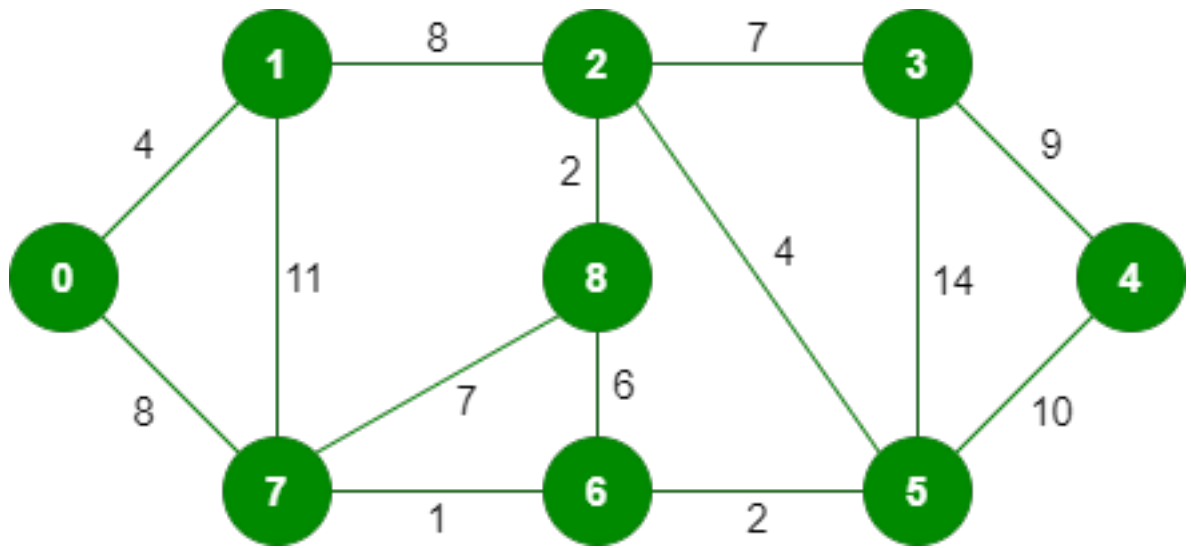
Steps for finding MST using Kruskal's algorithm:

%3. Sort all the edges in non-decreasing order of their weight.

%3. Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If cycle is not formed, include this edge. Else, discard it.

%3. Repeat step#2 until there are  $(V-1)$  edges in the spanning tree.

The algorithm is a Greedy Algorithm. The Greedy Choice is to pick the smallest weight edge that does not cause a cycle in the MST constructed so far. Let us understand it with an example: Consider the below input graph.



The graph contains 9 vertices and 14 edges. So, the minimum spanning tree formed will be having  $(9 - 1) = 8$  edges.

After sorting:

Weight	Src	Dest
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8



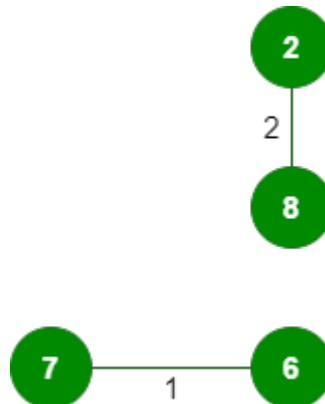
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

Now pick all edges one by one from the sorted list of edges

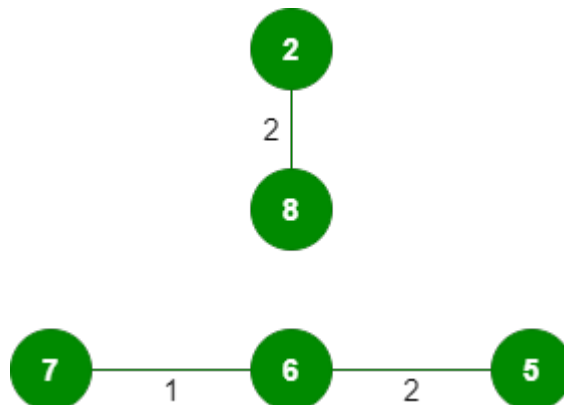
1. Pick edge 7-6: No cycle is formed, include it.



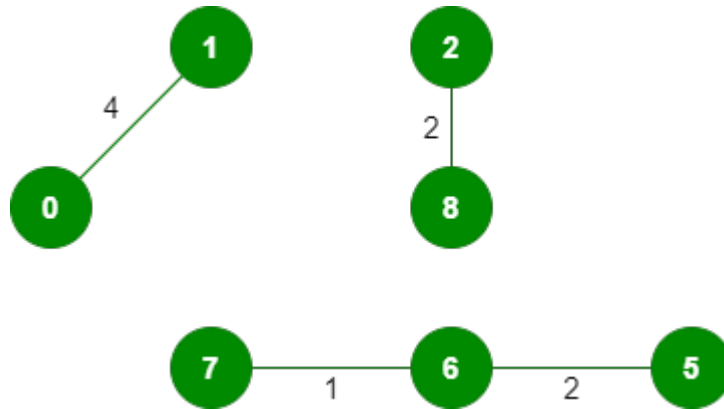
2. Pick edge 8-2: No cycle is formed, include it.



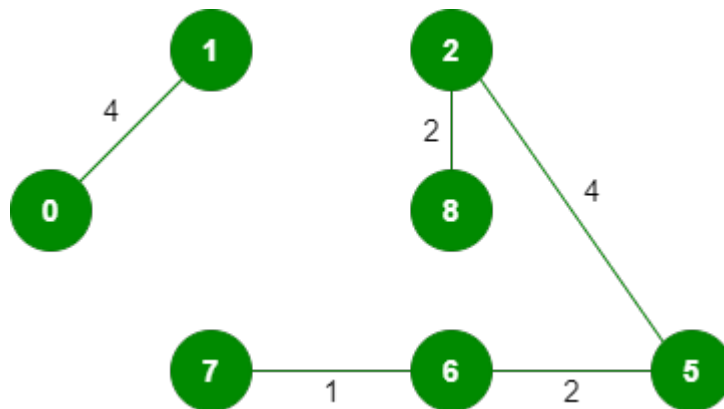
3. Pick edge 6-5: No cycle is formed, include it.



4. Pick edge 0-1: No cycle is formed, include it.

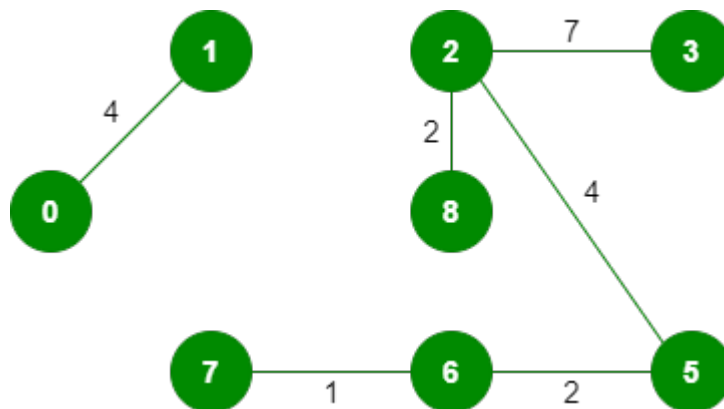


5. Pick edge 2-5: No cycle is formed, include it.



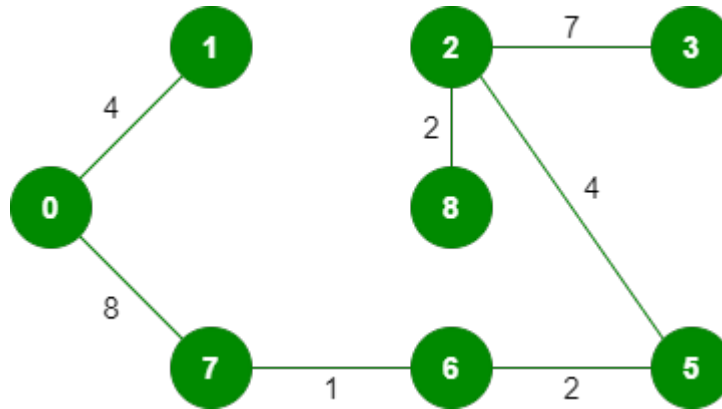
6. Pick edge 8-6: Since including this edge results in the cycle, discard it.

7. Pick edge 2-3: No cycle is formed, include it.



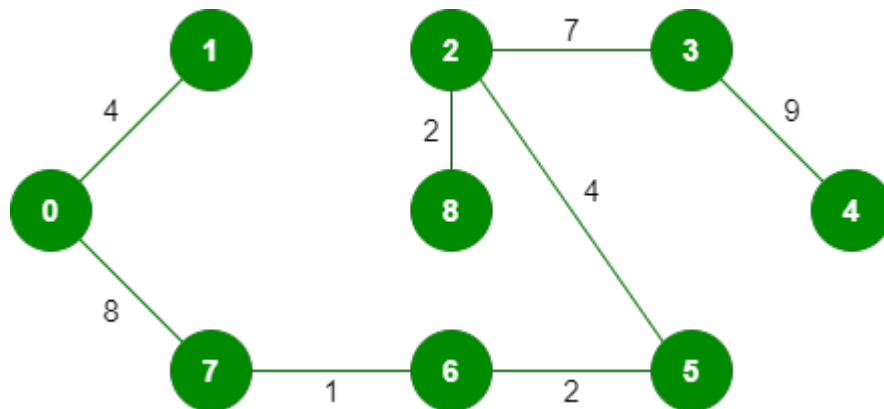
8. Pick edge 7-8: Since including this edge results in the cycle, discard it.

9. Pick edge 0-7: No cycle is formed, include it.



10. Pick edge 1-2: Since including this edge results in the cycle, discard it.

11. Pick edge 3-4: No cycle is formed, include it.



Since the number of edges included equals  $(V - 1)$ , the algorithm stops here.

### Prim's Minimum Spanning Tree Algorithm:

We have discussed Kruskal's algorithm for Minimum Spanning Tree. Like Kruskal's algorithm, Prim's algorithm is also a Greedy algorithm. It starts with an empty spanning tree. The idea is to maintain two sets of vertices. The first set contains the vertices already included in the MST, the other set contains the vertices not yet included. At every step, it considers all the edges that connect the two sets and picks the minimum weight edge from these edges. After picking the edge, it moves the other endpoint of the edge to the set containing MST.

A group of edges that connects two sets of vertices in a graph is called cut in graph theory. So, at every step of Prim's algorithm, we find a cut (of two sets, one contains

the vertices already included in MST and the other contains the rest of the vertices), pick the minimum weight edge from the cut, and include this vertex to MST Set (the set that contains already included vertices).

#### How does Prim's Algorithm Work?

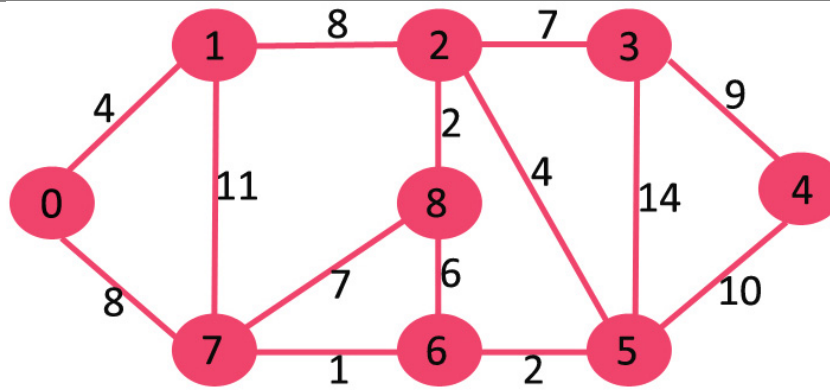
The idea behind Prim's algorithm is simple, a spanning tree means all vertices must be connected. So the two disjoint subsets (discussed above) of vertices must be connected to make a Spanning Tree. And they must be connected with the minimum weight edge to make it a Minimum Spanning Tree.

#### Algorithm:

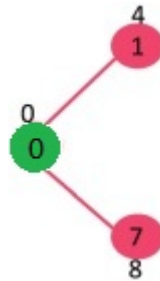
1. Create a set mstSet that keeps track of vertices already included in MST.
2. Assign a key value to all vertices in the input graph. Initialize all key values as INFINITE. Assign the key value as 0 for the first vertex so that it is picked first.
3. While mstSet doesn't include all vertices
  - a) Pick a vertex u which is not there in mstSet and has a minimum key value.
  - b) Include u to mstSet.
  - c) Update key value of all adjacent vertices of u. To update the key values, iterate through all adjacent vertices. For every adjacent vertex v, if the weight of edge u-v is less than the previous key value of v, update the key value as the weight of u-v

The idea of using key values is to pick the minimum weight edge from cut. The key values are used only for vertices that are not yet included in MST, the key value for these vertices indicates the minimum weight edges connecting them to the set of vertices included in MST.

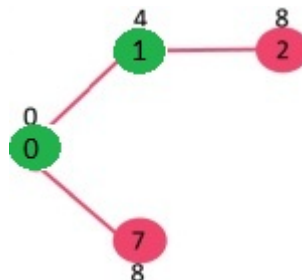
Let us understand with the following example:



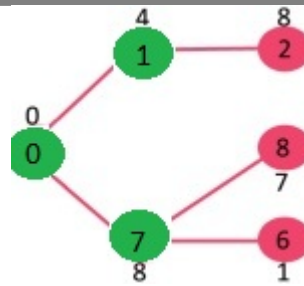
The set `mstSet` is initially empty and keys assigned to vertices are  $\{0, \text{INF}, \text{INF}, \text{INF}, \text{INF}, \text{INF}, \text{INF}, \text{INF}, \text{INF}\}$  where INF indicates infinite. Now pick the vertex with the minimum key value. The vertex 0 is picked, include it in `mstSet`. So `mstSet` becomes  $\{0\}$ . After including to `mstSet`, update key values of adjacent vertices. Adjacent vertices of 0 are 1 and 7. The key values of 1 and 7 are updated as 4 and 8. Following subgraph shows vertices and their key values, only the vertices with finite key values are shown. The vertices included in MST are shown in green color.



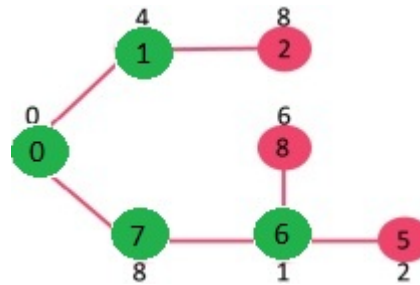
Pick the vertex with minimum key value and not already included in MST (not in `mstSet`). The vertex 1 is picked and added to `mstSet`. So `mstSet` now becomes  $\{0, 1\}$ . Update the key values of adjacent vertices of 1. The key value of vertex 2 becomes 8



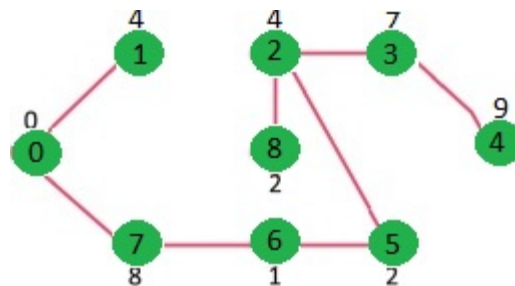
Pick the vertex with minimum key value and not already included in MST (not in `mstSet`). We can either pick vertex 7 or vertex 2, let vertex 7 is picked. So `mstSet` now becomes  $\{0, 1, 7\}$ . Update the key values of adjacent vertices of 7. The key value of vertex 6 and 8 becomes finite (1 and 7 respectively).



Pick the vertex with minimum key value and not already included in MST (not in mstSET). Vertex 6 is picked. So mstSet now becomes {0, 1, 7, 6}. Update the key values of adjacent vertices of 6. The key value of vertex 5 and 8 are updated.



We repeat the above steps until mstSet includes all vertices of given graph. Finally, we get the following graph.



**Conclusion:** Thus we have implemented Greedy search algorithm Prim's Minimal Spanning Tree Algorithm and Kruskal's Minimal Spanning Tree Algorithm.

<b>Lab Assignment No.</b>	13
<b>Title</b>	Develop an elementary chatbot for any suitable customer interaction application.
<b>Roll No.</b>	
<b>Class</b>	TE
<b>Date of Completion</b>	
<b>Subject</b>	Software Laboratory I
<b>Assessment Marks</b>	
<b>Assessor's Sign</b>	

## ASSIGNMENT No: 13

**Title:** Develop an elementary chatbot for any suitable customer interaction application.

**Problem Statement:** Develop an elementary chatbot for any suitable customer interaction application.

**Objective:**

Understand and Implement chatbot for any suitable customer interaction application

**Outcome:**

Ability to choose an appropriate problem solving method and knowledge representation technique

**Software Required:**

Python

**Theory:**

Understanding the Chatbot:

A Chatbot is an Artificial Intelligence-based software developed to interact with humans in their natural languages. These chatbots are generally converse through auditory or textual methods, and they can effortlessly mimic human languages to communicate with human beings in a human-like way. A chatbot is considered one of the best applications of natural languages processing.

Two category the Chatbots:

Rule-based Chatbots:

The Rule-based approach trains a chatbot to answer questions based on a list of pre-determined rules on which it was primarily trained. These set rules can either be pretty simple or quite complex, and we can use these rule-based chatbots to handle simple queries but not process more complicated requests or queries.

Self-learning Chatbots:

Self-learning chatbots are chatbots that can learn on their own. These leverage advanced technologies such as Artificial Intelligence (AI) and Machine Learning (ML) to train themselves from behaviours and instances. Generally, these chatbots are quite smarter than rule-based bots. We can classify the Self-learning chatbots furtherly into two categories - Retrieval-based Chatbots and Generative Chatbots.



### Retrieval-based Chatbots:

A retrieval-based chatbot works on pre-defined input patterns and sets responses. Once the question or pattern is inserted, the chatbot utilizes a heuristic approach to deliver the relevant response. The model based on retrieval is extensively utilized to design and develop goal-oriented chatbots using customized features such as the flow and tone of the bot in order to enhance the experience of the customer.

### Generative Chatbots:

Unlike retrieval-based chatbots, generative chatbots are not based on pre-defined responses - they leverage seq2seq neural networks. This is constructed on the concept of machine translation, where the source code is converted from one language to another language. In the seq2seq approach, the input is changed into an output.

The first chatbot named ELIZA was designed and developed by Joseph Weizenbaum in 1966 that could imitate the language of a psychotherapist in only 200 lines of code. But as the technology gets more advance, we have come a long way from scripted chatbots to chatbots in Python today.

### Chatbot in present Generation:

Today, we have smart Chatbots powered by Artificial Intelligence that utilize natural language processing (NLP) in order to understand the commands from humans (text and voice) and learn from experience. Chatbots have become a staple customer interaction utility for companies and brands that have an active online existence (website and social network platforms).

With the help of Python, Chatbots are considered a nifty utility as they facilitate rapid messaging between the brand and the customer. Let us think about Microsoft's Cortana, Amazon's Alexa, and Apple's Siri. Aren't these chatbots wonderful? It becomes quite interesting to learn how to create a chatbot using the Python programming language.

Fundamentally, the chatbot utilizing Python is designed and programmed to take in the data we provide and then analyze it using the complex algorithms for Artificial Intelligence. It then delivers us either a written response or a verbal one. Since these bots can learn from experiences and behavior, they can respond to a large variety of queries and commands.

Although chatbot in Python has already started to rule the tech scenario at present, chatbots had handled approximately 85% of the customer-brand interactions by 2020 as per the prediction of Gartner.

In light of the increasing popularity and adoption of chatbots in the industry, we can increase the market value by learning how to create a chatbot in Python - among the most extensively utilized programming languages globally.

### Understanding the ChatterBot Library:

ChatterBot is a Python library that is developed to provide automated responses to user

inputs. It makes utilization of a combination of Machine Learning algorithms in order to generate multiple types of responses. This feature enables developers to construct chatbots using Python that can communicate with humans and provide relevant and appropriate responses. Moreover, the ML algorithms support the bot to improve its performance with experience.

Another amazing feature of the ChatterBot library is its language independence. The library is developed in such a manner that makes it possible to train the bot in more than one programming language.

### Understanding the working of the ChatterBot library:

When a user inserts a particular input in the chatbot (designed on ChatterBot), the bot saves the input and the response for any future usage. This information (of gathered experiences) allows the chatbot to generate automated responses every time a new input is fed into it.

The program picks the most appropriate response from the nearest statement that matches the input and then delivers a response from the already known choice of statements and responses. Over time, as the chatbot indulges in more communications, the precision of reply progresses.

### Creating a Chatbot using Python:

We will follow a step-by-step approach and break down the procedure of creating a Python chat.

We will begin building a Python chatbot by importing all the required packages and modules necessary for the project. We will also initialize different variables that we want to use in it. Moreover, we will also be dealing with text data, so we have to perform data preprocessing on the dataset before designing an ML model.

This is where tokenizing supports text data - it converts the large text dataset into smaller, readable chunks (such as words). Once this process is complete, we can go for lemmatization to transform a word into its lemma form. Then it generates a pickle file in order to store the objects of Python that are utilized to predict the responses of the bot.

Another major section of the chatbot development procedure is developing the training and testing datasets.

### Why Chatbots are important for a Business or a Website:

- Quick resolution for a complaint or a problem.

- Improve business branding thereby achieving great customer satisfaction.

- Answering questions and answers for customers.

- Making a reservation at hotel or at restaurant.

- Save human effort 24x7.

Enhance business revenue by providing ideas and inspirations.

Finding details about business such as hours of operation, phone number and address.

Automate sales and lead generation process.

Reduce customer agents waiting time answering phone calls.

#### Benefits of using Chatbots:

24x7 availability.

Instant answers to queries.

Support multi-language to enhance businesses.

Simple and Easy to Use UI to engage more customers.

Cost effective and user interactive.

Avoid communication with call agents thereby reducing the time consuming tasks.

Understand the Customer behavior

Increase sales of business by offering promo codes or gifts.

**Conclusion:** Thus we have developed an elementary chatbot for any suitable customer interaction application.

<b>Lab Assignment No.</b>	14
<b>Title</b>	Mini Project: Implement any one of the following Expert System Information management Hospitals and medical facilities Help desks management Employee performance evaluation Stock market trading Airline scheduling and cargo schedules
<b>Roll No.</b>	
<b>Class</b>	TE
<b>Date of Completion</b>	
<b>Subject</b>	Software Laboratory I
<b>Assessment Marks</b>	
<b>Assessor's Sign</b>	

## ASSIGNMENT No: 14

**Title:** Implement any one of the following Expert System

Information management

Hospitals and medical facilities

Help desks management

Employee performance evaluation

Stock market trading

Airline scheduling and cargo schedules

**Problem Statement:** Implement any one of the following Expert System

Information management

Hospitals and medical facilities

Help desks management

Employee performance evaluation

Stock market trading

Airline scheduling and cargo schedules

**Objective:**

Understand and implement Expert System

**Outcome:**

Ability to choose an appropriate problem solving method and knowledge representation technique

**Software Required:**

Python

**Theory:**

Expert Systems:

Artificial Intelligence is a piece of software that simulates the behaviour and judgement of a human or an organization that has experts in a particular domain is known as an expert system. It does this by acquiring relevant knowledge from its knowledge base and interpreting it according to the user's problem. The data in the knowledge base is added by humans that are expert in a particular domain and this software is used by a non-expert user to acquire some information. It is widely used in many areas such as medical diagnosis, accounting, coding, games etc.

An expert system is AI software that uses knowledge stored in a knowledge base to solve problems that would usually require a human expert thus preserving a human expert's knowledge in its knowledge base. They can advise users as well as provide explanations to them about how they reached a particular conclusion or advice. Knowledge Engineering is the term used to define the process of building an Expert System and its practitioners are called Knowledge Engineers. The primary role of a knowledge engineer is to make sure that the computer possesses all the knowledge required to solve a problem. The knowledge engineer must choose one or more forms in which to represent the required knowledge as a symbolic pattern in the memory of the computer.

### Example :

There are many examples of an expert system. Some of them are given below –

#### MYCIN –

One of the earliest expert systems based on backward chaining. It can identify various bacteria that can cause severe infections and can also recommend drugs based on the person's weight.

#### DENDRAL –

It was an artificial intelligence-based expert system used for chemical analysis. It used a substance's spectrographic data to predict its molecular structure.

#### R1/XCON –

It could select specific software to generate a computer system wished by the user.

#### PXDES –

It could easily determine the type and the degree of lung cancer in a patient based on the data.

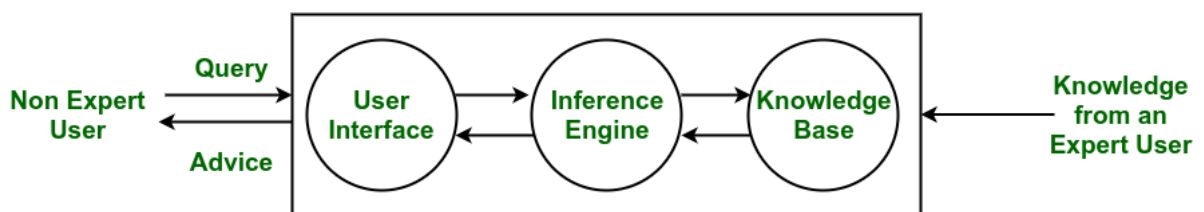
#### CaDet –

It is a clinical support system that could identify cancer in its early stages in patients.

#### DXplain –

It was also a clinical support system that could suggest a variety of diseases based on the findings of the doctor.

### Components of an Expert System:



Architecture of an Expert System

#### Knowledge Base –

The knowledge base represents facts and rules. It consists of knowledge in a particular domain as well as rules to solve a problem, procedures and intrinsic data relevant to the domain.

#### Inference Engine –

The function of the inference engine is to fetch the relevant knowledge from the

knowledge base, interpret it and to find a solution relevant to the user's problem. The inference engine acquires the rules from its knowledge base and applies them to the known facts to infer new facts. Inference engines can also include an explanation and debugging abilities.

#### Knowledge Acquisition and Learning Module –

The function of this component is to allow the expert system to acquire more and more knowledge from various sources and store it in the knowledge base.

#### User Interface –

This module makes it possible for a non-expert user to interact with the expert system and find a solution to the problem.

#### Explanation Module –

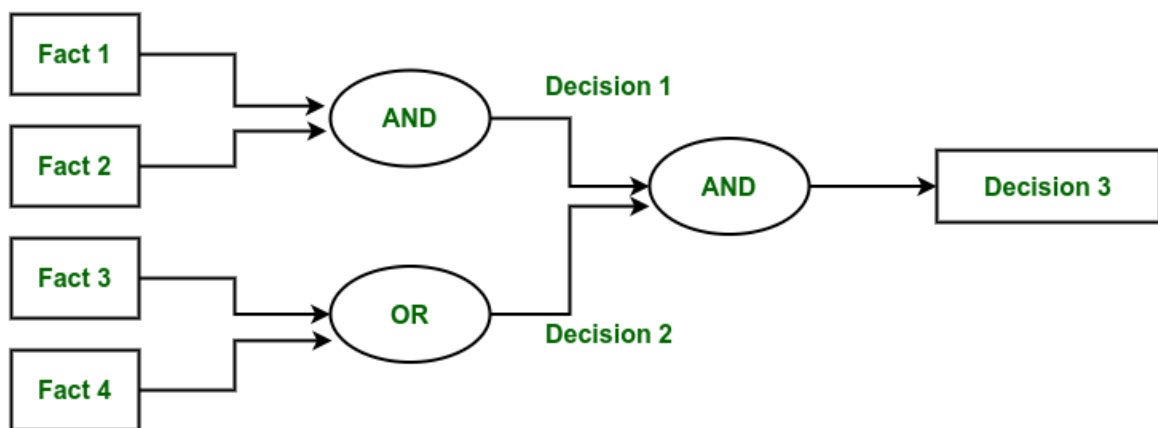
This module helps the expert system to give the user an explanation about how the expert system reached a particular conclusion.

### Two strategies:

The Inference Engine generally uses two strategies for acquiring knowledge from the Knowledge Base, namely –

#### Forward Chaining:

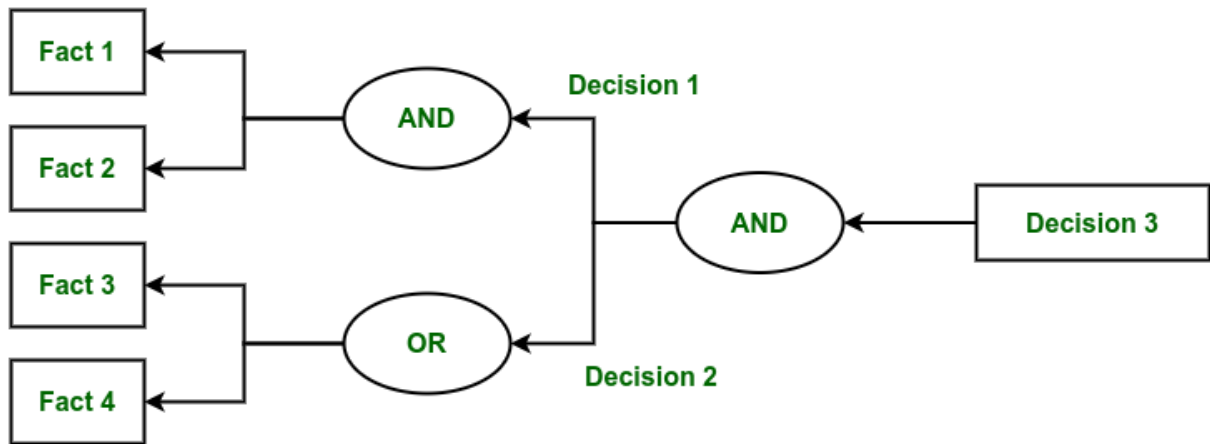
It is a strategic process used by the Expert System to answer the questions – What will happen next. This strategy is mostly used for managing tasks like creating a conclusion, result or effect. Example – prediction or share market movement status.



#### Forward Chaining

#### Backward Chaining

It is a storage used by the Expert System to answer the questions – Why this has happened. This strategy is mostly used to find out the root cause or reason behind it, considering what has already happened. Example – diagnosis of stomach pain, blood cancer or dengue, etc.



### Backward Chaining

#### Characteristics of an Expert System:

Human experts are perishable, but an expert system is permanent.

It helps to distribute the expertise of a human.

One expert system may contain knowledge from more than one human experts thus making the solutions more efficient.

It decreases the cost of consulting an expert for various domains such as medical diagnosis.

They use a knowledge base and inference engine.

Expert systems can solve complex problems by deducing new facts through existing facts of knowledge, represented mostly as if-then rules rather than through conventional procedural code.

Expert systems were among the first truly successful forms of artificial intelligence (AI) software.

#### Limitations:

Do not have human-like decision-making power.

Cannot possess human capabilities.

Cannot produce correct result from less amount of knowledge.

Requires excessive training.

#### Advantages:

Low accessibility cost.



Fast response.

Not affected by emotions, unlike humans.

Low error rate.

Capable of explaining how they reached a solution.

### Disadvantages:

The expert system has no emotions.

Common sense is the main issue of the expert system.

It is developed for a specific domain.

It needs to be updated manually. It does not learn itself.

Not capable to explain the logic behind the decision.

### Applications:

The application of an expert system can be found in almost all areas of business or government. They include areas such as –

Different types of medical diagnosis like internal medicine, blood diseases and show on.

Diagnosis of the complex electronic and electromechanical system.

Diagnosis of a software development project.

Planning experiment in biology, chemistry and molecular genetics.

Forecasting crop damage.

Diagnosis of the diesel-electric locomotive system.

Identification of chemical compound structure.

Scheduling of customer order, computer resources and various manufacturing task.

Assessment of geologic structure from dip meter logs.

Assessment of space structure through satellite and robot.

The design of VLSI system.

Teaching students specialize task.

Assessment of log including civil case evaluation, product liability etc.

Expert systems have evolved so much that they have started various debates about the fate of humanity in the face of such intelligence, with authors such as Nick Bostrom (Professor of Philosophy at Oxford University), pondering if computing power has transcended our ability to control it.

**Conclusion:** Thus we have implemented Expert System

# Group C

Lab Assignment No.	Mini Project
Title	Develop application considering: Front End: Python/Java/PHP/Perl/Ruby/.NET/ or any other language Backend : MongoDB/ MySQL/ Oracle / or any standard SQL / NoSQL databas
Roll No.	
Class	TE

<b>Date of Completion</b>	
<b>Subject</b>	Software Laboratory I
<b>Assessment Marks</b>	
<b>Assessor's Sign</b>	

## [DBMS]Mini Project

**Title:** Develop application considering:

Front End: Python/Java/PHP/Perl/Ruby/.NET/ or any other language

Backend : MongoDB/ MySQL/ Oracle / or any standard SQL / NoSQL database

### Problem Statement:

Develop an application with following details:

1. Follow the same problem statement decided in Assignment-1 of Group A.
2. Follow the Software Development Life cycle and other concepts learnt in Software Engineering Course throughout the implementation.
3. Develop application considering:  
Front End: Python/Java/PHP/Perl/Ruby/.NET/ or any other language  
Backend : MongoDB/ MySQL/ Oracle / or any standard SQL / NoSQL database
4. Test and validate application using Manual/Automation testing.
5. Student should develop application in group of 2-3 students and submit the Project Report which will consist of documentation related to different phases of Software Development Life Cycle:  
Title of the Project, Abstract, Introduction  
Software Requirement Specification (SRS)  
Conceptual Design using ER features, Relational Model in appropriate Normalize form  
Graphical User Interface, Source Code  
Testing document  
Conclusion

**Objective:** Ability to develop an application.

**Outcome:**

Able to develop an application using front and any backend by following software development lifecycle

**Software Required:**

Python/Java , MongoDB/MySQL/Oracle

**Guidelines:**

Sr No.	Assignment	Remark
1	Write an Abstract of Application decided by you & approved by me	
2	Construct an ER Diagram	
3	Convert ER-Diagram into Tables	
4	Normalize all the tables into at-least in BCNF	
5	Write an SQL (DDL) Statements to create all tables	
6	Perform all SQL (DML-Select, Insert, Delete, Update) statements	
7	Create Php form for Login Details	
8	Create a Project	