



K. K. WAGH INSTITUTE OF ENGINEERING EDUCATION AND RESEARCH, NASIK

ARTIFICIAL INTELLIGENCE & DATA SCIENCE DEPARTMENT

2023-2024

LABORATORY MANUAL

Software Laboratory I

TE-AI & DS

SEMESTER-V

TEACHING SCHEME

Practical: 04 Hrs/Week

EXAMINATION SCHEME

Term Work: 25 Marks

Practical: 25 Marks



K.K. Wagh Institute of Engineering Education and Research, Nashik

Vision

Empowering through Quality Technical Education

Mission

Committed to serving the needs of the society at large by imparting state-of-the-art Engineering education and to provide knowledge and developing Attitudes, Skills and Values leading to establishment of quality conscious and sustainable research oriented Educational Institute

Department of Artificial Intelligence & Data Science

Vision

To be a valuable resource for industry and society through quality education and research in Artificial Intelligence and Data Science

Mission

1. To impart knowledge and inculcate skills by nurturing a conducive learning environment
2. To promote research and development in collaboration with industry
3. To build conducive environment for R&D-based innovation to serve the emerging needs of society
4. To develop attitude and inculcate values for character building and holistic development

Programme Educational Objectives (PEOs)

1. To develop core competencies in the field of Artificial Intelligence & Data Science
2. To teach professional skills and ethics
3. To develop an ability to conduct research

Programme Specific Outcomes (PSOs)

Engineering Graduates will be able to:

1. Interpret data and apply Artificial Intelligence algorithms to provide solutions
2. Apply standard practices, and strategies and use appropriate models of data science to discover knowledge

Program Outcomes

Students will be able to:

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems.
2. Problem analysis: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.
4. Conduct investigations of complex problems: Use research based knowledge and research methods including design of experiments, analysis & interpretation of data and synthesis of the information to provide valid conclusions.
5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools, including prediction and modeling to complex engineering activities, with an understanding of the limitations.
6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. Individual and team work: Function effectively as an individual, and as member or leader in diverse teams, and in multidisciplinary settings.
10. Communication: Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

LIST OF ASSIGNMENTS

Subject: Software Laboratory I

Sr · N	TITLE	Mappin g With
Group A		
1	<p>SQL Queries:</p> <ul style="list-style-type: none"> Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym, different constraints etc. Write at least 10 SQL queries on the suitable database application using SQL DML statements. <p>Note: Instructor will design the queries which demonstrate the use of concepts like Insert, Select, Update, Delete with operators, functions, and set operator etc.</p> <p>Consider the given Database Schema: for problem statement 1</p> <ol style="list-style-type: none"> Dept (<u>Deptno</u> , Name , Location, Managerempid) Employees (name, <u>empid</u>, address, city, dob, date_of_joining,gender, salary, deptno) <p>Gender must take value ‘M’ or ‘F’.</p> <ol style="list-style-type: none"> Project(<u>Projectid</u>, title,city). Works (empid , Projectid, total_hrs_worked); Dependant(empid, name_of_dependant, age, relation) <p><i>Primary Key is underlined.</i></p> <p>Projects are „Banking Project“, „Testing Project“, „ERP system“. „Software management“ etc</p> <p>Consider departments as””Development” . “Testing”, “Planning”, “Service”, “Administration” etc</p> <p>Consider Locations for department as “Building 1”, “Building 2”, “Building 3”, “Main building”</p> <p>employee id will start as 101 to 300</p>	CO1
2	<p>SQL Queries all types of Join, Sub-Query and View:</p> <p>Write at least10 SQL queries for suitable database applications using SQL DML statements.</p> <p>Note: Instructor will design the queries which demonstrate the use of concepts like all types of Join ,Sub-Query and View</p>	CO1
3	Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators etc.).	CO2

Subject Code: 317523

4	<p>Unnamed PL/SQLcode block: Use of Control structure and Exception handling is mandatory.</p> <p>Suggested Problem statement:</p> <ol style="list-style-type: none"> Consider Tables: <ol style="list-style-type: none"> Borrower(Roll_no, Name, Date of Issue, Name of Book, Status) Fine(Roll_no, Date, Amt) <ol style="list-style-type: none"> Accept Roll_no and Name of Book from the user. Check the number of days (from date of issue). If days are between 15 to 30 then the fine amount will be Rs 5per day. If no. of days>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 <p>Also handles the exception by named exception handler or user defined exception handler</p>	CO1
5	<p>Exporting and Importing data</p> <ul style="list-style-type: none"> Design and develop SQL DML statements to demonstrate exporting tables to external files of different file formats ex. CSV, XLSX, TXT, etc. <p>Design and develop SQL DML statements to demonstrate importing data from external files of different file formats ex. CSV, XLSX, TXT, etc.</p>	CO3
6	<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, edit etc.)</p>	CO3
Group B		
7	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.	CO4
8	Implement n-queens problem.	CO4
9	Implement Alpha-Beta Tree search for any game search problem.	CO5
10	<p>Implement Greedy search algorithm for any of the following application:</p> <ul style="list-style-type: none"> 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 	CO4
11	Develop an elementary chatbot for any suitable customer interaction application.	CO6
12	<p>Mini Project: Implement any one of the following Expert System</p> <ul style="list-style-type: none"> Information management Hospitals and medical facilities Help desks management Employee performance evaluation Stock market trading Airline scheduling and cargo schedules 	CO6

Group C		
13	<p>Develop an application with following details:</p> <ol style="list-style-type: none"> 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: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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. <p>Note: Instructor should maintain progress report of mini project throughout the semester from project group.</p>	CO1, CO2, CO3

Assignment No. 1

Title: SQL Queries:

Consider the given Database Schema: for problem statement 1

1. Dept (Deptno , Name , Location, Managerempid)
2. Employees (name, empid, address, city, dob, date_of_joining,gender, salary, deptno)
Gender must take value 'M' or 'F'.
3. Project(Projectid, title,city).
4. Works (empid , Projectid, total_hrs_worked);
5. Dependant(empid, name_of_dependant, age, relation)

Primary Key is underlined.

Projects are „Banking Project“, „Testing Project“, „ERP system“. „Software management“ etc

Consider departments as””Development” . “Testing”, “Planning”, “Service”, “Administration” etc

Consider Locations for department as “Building 1”, “Building 2”, “Building 3”, “Main building”

employee id will start as 101 to 300

Objectives: To learn how Write SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operator.

Outcomes: Student will be able to write SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operator

Theory- Concept in brief :

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.

IBM developed the original version of SQL, originally called Sequel, as part of the System R project in the early 1970s. The Sequel language has evolved since then, and its name has changed to SQL (Structured Query language).

FEATURES OF SQL:

1. Scalability and Flexibility: SQL provide Scalability and Flexibility. It is very easy to create new tables and previously created or not used tables can be dropped or deleted in a database.
2. Robust Transactional Support: With SQL programming can handle large records and manage numerous transactions.
3. High Security: It is very easy to provide permissions on tables, procedures, and views hence SQL give security to your data.
4. Comprehensive Application Development: SQL is used by many programmers to program apps to access a database. No matter what is the size of organization, SQL works for every small or large organization.

5.Management Ease: SQL is used in almost every relational database management system. “Select”, “Create”, “Insert”, “Drop”, “Update”, and “Delete” are the standard and common SQL commands that helps us to manage large amount of data from a database very quickly and efficiently.

5.Open Source: SQL is an open-source programming language for building relational database management system

SQL Process

When you are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task.

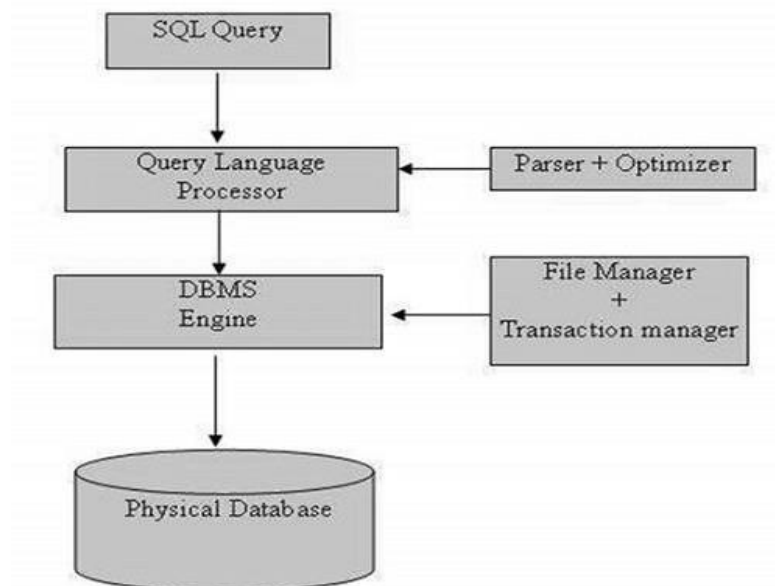
There are various components included in this process.

These components are –

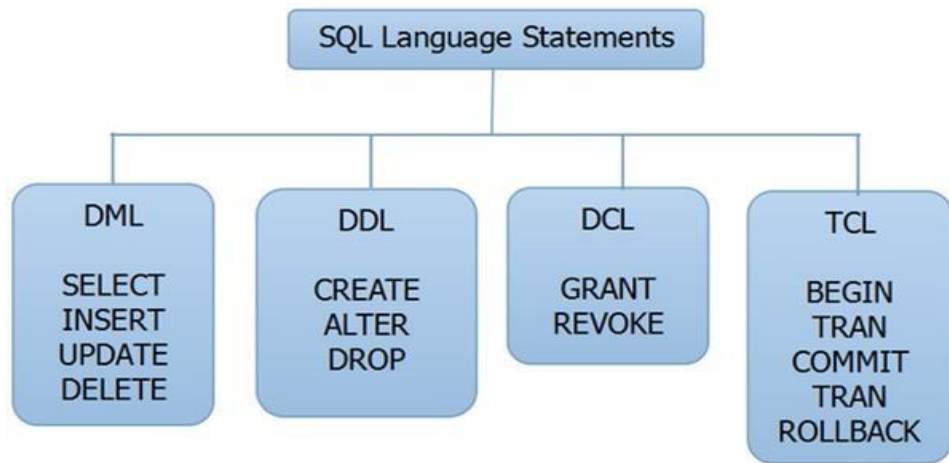
- Query Dispatcher
- Optimization Engines
- Classic Query Engine
- SQL Query Engine, etc.

A classic query engine handles all the non-SQL queries, but a SQL query engine won't handle logical files.

Following is a simple diagram showing the SQL Architecture –



TYPES OF SOL COMMANDS:



DDL - Data Definition Language

Sr.No.	Command & Description										
1	<p>CREATE :Creates a new table, a view of a table, or other object in the database. Syntax</p> <pre>CREATE TABLE table_name(column1 datatype, column2 datatype, column3 datatype, columnN datatype, [PRIMARY KEY(one or more columns),]);</pre> <p>e.g.</p> <pre>CREATE TABLE Persons (</pre> <table><tr><td>PersonID</td><td>int,</td></tr><tr><td>LastName</td><td>varchar(55),</td></tr><tr><td>FirstName</td><td>varchar(55),</td></tr><tr><td>Address</td><td>varchar(255),</td></tr><tr><td>City</td><td>varchar(55)</td></tr></table> <pre>);</pre>	PersonID	int,	LastName	varchar(55),	FirstName	varchar(55),	Address	varchar(255),	City	varchar(55)
PersonID	int,										
LastName	varchar(55),										
FirstName	varchar(55),										
Address	varchar(255),										
City	varchar(55)										
2	<p>ALTER: Modifies an existing database object, such as a table. Syntax</p> <pre>ALTER TABLE table_name RENAME TO new_table_name; ALTER TABLE table_name ADD column_name datatype; ALTER TABLE table_name DROP COLUMN column_name; ALTER TABLE table_name MODIFY COLUMN column_name datatype;</pre> <p>e.g</p> <pre>ALTER TABLE Customers ADD Email varchar(255);</pre>										

3	DROP : Deletes an entire table, a view of a table or other objects in the database. Syntax
	DROP TABLE table_name;
	e.g.
	DROP TABLE Customers;

DML - Data Manipulation Language

Sr.No.	Command & Description
1	SELECT: Retrieves certain records from one or more tables. Syntax SELECT column-names FROM table-name [WHERE condition] E.g. SELECT CustomerName, City FROM Customers; select * from student; – Get all the records of student table
2	INSERT INTO : Creates a record. Syntax INSERT INTO table-name (column-names) VALUES (values) OR INSERT INTO table-name (column-names) SELECT column-names FROM table-name WHERE condition; e.g. INSERT INTO Customer (FirstName, LastName, City, Country, Phone) VALUES ('Craig', 'Smith', 'New York', 'USA', 1-01-993 2800);
3	UPDATE : Modifies records. Syntax UPDATE table_name SET column1 = value1, column2 = value2. ..columnN=valueN [WHERE CONDITION]; E.g. UPDATE Customers SET ContactName = 'Alfred Schmidt', City= 'Frankfurt' WHERE CustomerID = 1;
4	DELETE : Deletes records.

	Syntax
	DELETE FROM table_name WHERE {CONDITION};
	e.g.
	DELETE FROM Customers WHERE CustomerName='Nitin Shah';

DCL - Data Control Language

Sr.No.	Command & Description
1	GRANT Gives a privilege to user. GRANT privilege_name ON object_name TO {user_name PUBLIC role_name} [WITH GRANT OPTION]; e.g. GRANT SELECT, INSERT, UPDATE, DELETE ON employees TO smithj;
2	REVOKE Takes back privileges granted from user. REVOKE privilege_name ON object_name FROM {user_name PUBLIC role_name} e.g. REVOKE DELETE ON employees FROM anderson;

TCL - Transaction Control Language

The commands are used to manage the transactions in the database. These are used to manage the changes made by DML statements.

Sr.No.	Command & Description
1	COMMIT Commit command is used to permanently save any transaction into the database. Syntax: Commit;
2	ROLLBACK Rollback command is used to restore the database for the last committed state. It's also used with save point to jump to the save point. Syntax: rollback;

Assignment No. 2

Title: SQL Queries (all types of Join, Sub-Query and View)

Write at least 10 SQL queries for suitable database applications 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.

1. Develop a SQL query to list employees having birthday in Jan .
2. Develop a SQL query to Find the names of all employees who work for “ERP project
3. Develop a SQL query to Find all employees in the database who live in the same cities as the project for which they work.
4. Develop a SQL query to find average salary of each department
5. Develop a SQL query to Find the department that has the most employees.
6. Develop a SQL query to Find the department that has the smallest payroll.
7. Develop a SQL query to Find the employees working on each project.
8. Develop a SQL query to Find the employees who are not having any project.
9. Develop a SQL query to Find the employees whose department is located in “main building”
10. Create a view containing the total number of employees whose project location is “Pune”

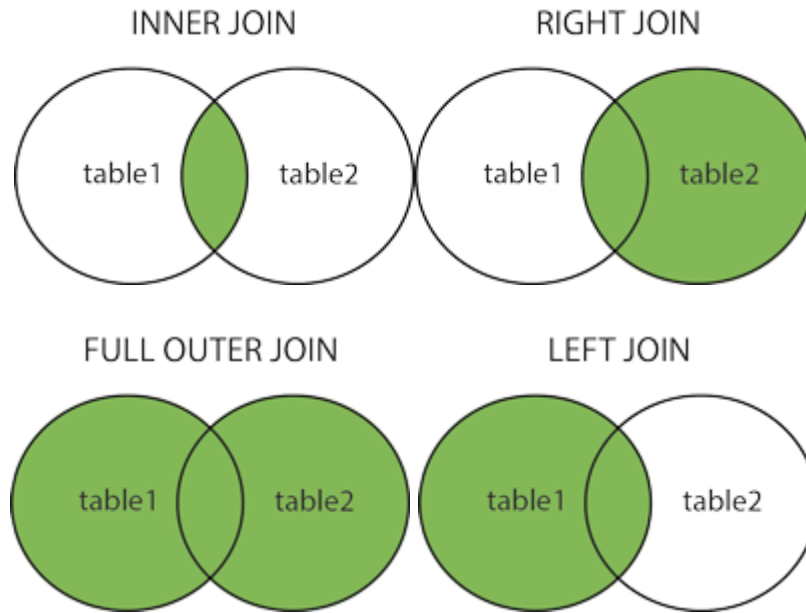
Objectives: To create logical design of database for a given application and formulate adequate and efficient database queries using joins and aggregate and create views

Outcomes: Student will be able to create logical design of database for a given application and formulate adequate and efficient database queries using joins and aggregate and create views

Theory- Concept in brief :

Different Types of SQL JOINS:

- **(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



Inner Join Syntax:

```
SELECT column_name(s) FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

Left Join Syntax:

```
SELECT column_name(s) FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name;
```

Right Join Syntax:

```
SELECT column_name(s) FROM table1
RIGHT JOIN table2 ON table1.column_name = table2.column_name;
```

Full Outer Join Syntax:

```
SELECT column_name(s) FROM table1
FULL OUTER JOIN table2 ON table1.column_name = table2.column_name;
```

Example :

```
mysql> SELECT * FROM STUDENT;
+-----+-----+-----+-----+
| STUDENT_ID | STUDENT_NAME | INSTRUCTOR_ID | STUDENT_CITY |
+-----+-----+-----+-----+
| 11 | VRUSHALI | I-101 | NASHIK |
| 15 | SNEHAL | I-105 | PUNE |
| 16 | POOJA | I-109 | NASHIK |
| 17 | ANKITA | I-112 | MUMBAI |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM INSTRUCTOR;
```

INSTRUCTOR_ID	INSTRUCTOR_NAME	INSTRUCTOR_CITY	SPECIALIZATION
I-101	XYZ	PUNE	MECHANICAL ENGG
I-105	ABC	NASHIK	COMPUTER
I-106	PQR	MUMBAI	COMPUTER
I-112	JKL	PUNE	CIVIL

4 rows in set (0.00 sec)

1: Find the instructor of each student.

```
mysql> SELECT STUDENT_NAME,INSTRUCTOR_NAME FROM STUDENT NATURAL JOIN INSTRUCTOR;
```

STUDENT_NAME	INSTRUCTOR_NAME
VRUSHALI	XYZ
SNEHAL	ABC
ANKITA	JKL

3 rows in set (0.02 sec)

OR

```
mysql> SELECT STUDENT_NAME,INSTRUCTOR_NAME FROM STUDENT INNER JOIN INSTRUCTOR ON STUDENT.INSTRUCTOR_ID=INSTRUCTOR.INSTRUCTOR_ID;
```

STUDENT_NAME	INSTRUCTOR_NAME
VRUSHALI	XYZ
SNEHAL	ABC
ANKITA	JKL

3 rows in set (0.00 sec)

Aggregation:

An aggregate function in SQL returns one value after calculating multiple values of a column. We often use aggregate functions with the GROUP BY and HAVING clauses of the SELECT statement.

Various types of SQL aggregate functions are:

- Count()
- Sum()
- Avg()
- Min()
- Max()

COUNT() Function

The COUNT() function returns the number of rows in a database table.

Syntax:

COUNT(*)

or

COUNT([ALL|DISTINCT] expression)

Id Name Salary

1 A 80

2 B 40

3 C 60

4 D 70

5 E 60

6 F Null

Count()*: Returns total number of records .i.e 6.

Count(salary): Return number of Non Null values over the column salary. i.e 5.

Count(Distinct Salary): Return number of distinct Non Null values over the column salary .i.e 4

Sum():

sum(salary): Sum all Non Null values of Column salary i.e., 310

sum(Distinct salary): Sum of all distinct Non-Null values i.e., 250.

Avg():

Avg(salary) = Sum(salary) / count(salary) = 310/5

Avg(Distinct salary) = sum(Distinct salary) / Count(Distinct Salary) = 250/4

Min():

Min(salary): Minimum value in the salary column except NULL i.e., 40.

Max(salary): Maximum value in the salary i.e., 80.

SQL | GROUP BY

The GROUP BY Statement in SQL is used to arrange identical data into groups with the help of some functions. i.e if a particular column has the same values in different rows then it will arrange these rows in a group.

Important Points:

- GROUP BY clause is used with the SELECT statement.
- In the query, GROUP BY clause is placed after the WHERE clause.
- In the query, GROUP BY clause is placed before ORDER BY clause if used any.

Syntax:

SELECT column1, function_name(column2)

FROM table_name

WHERE condition

GROUP BY column1, column2

ORDER BY column1, column2;

function_name: Name of the function used for example, SUM() , AVG().

table_name: Name of the table.

condition: Condition used.

Sample Table:

Employee

SI NO	NAME	SALARY	AGE
1	Harsh	2000	19
2	Dhanraj	3000	20
3	Ashish	1500	19
4	Harsh	3500	19
5	Ashish	1500	19

Example:

Group By single column: Group By single column means, to place all the rows with same value of only that particular column in one group. Consider the query as shown below:

```
SELECT NAME, SUM(SALARY) FROM Employee  
GROUP BY NAME;
```

- The above query will produce the below output:

NAME	SALARY
Ashish	3000
Dhanraj	3000
Harsh	5500

Student

SUBJECT	YEAR	NAME
English	1	Harsh
English	1	Pratik
English	1	Ramesh
English	2	Ashish
English	2	Suresh
Mathematics	1	Deepak
Mathematics	1	Sayan

Group By multiple columns: Group by multiple column is say for example, GROUP BY column1, column2. This means to place all the rows with same values of both the columns column1 and column2 in one group.

Consider the below query:

```
SELECT SUBJECT, YEAR, Count(*)  
FROM Student  
GROUP BY SUBJECT, YEAR;
```

Output:

SUBJECT	YEAR	Count
English	1	3
English	2	2
Mathematics	1	2

HAVING Clause

We know that WHERE clause is used to place conditions on columns but what if we want to place conditions on groups?

This is where the HAVING clause comes into use. We can use HAVING clauses to place conditions to decide which group will be the part of the final result-set. Also we can not use the aggregate functions like SUM(), COUNT() etc. with WHERE clauses. So we have to use the HAVING clause if we want to use any of these functions in the conditions.

Syntax:

```
SELECT column1, function_name(column2)  
FROM table_name  
WHERE condition  
GROUP BY column1, column2  
HAVING condition  
ORDER BY column1, column2;
```

function_name: Name of the function used for example, SUM() , AVG().

table_name: Name of the table.

condition: Condition used.

Example:

```
SELECT NAME, SUM(SALARY) FROM Employee  
GROUP BY NAME  
HAVING SUM(SALARY)>3000;
```

Output:

NAME	SUM(SALARY)
HARSH	5500

View:

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;
```

Assignment No. 3

Title: Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators etc.).

Objectives: To learn how Write MongoDB queries for suitable database application using CRUD operations.

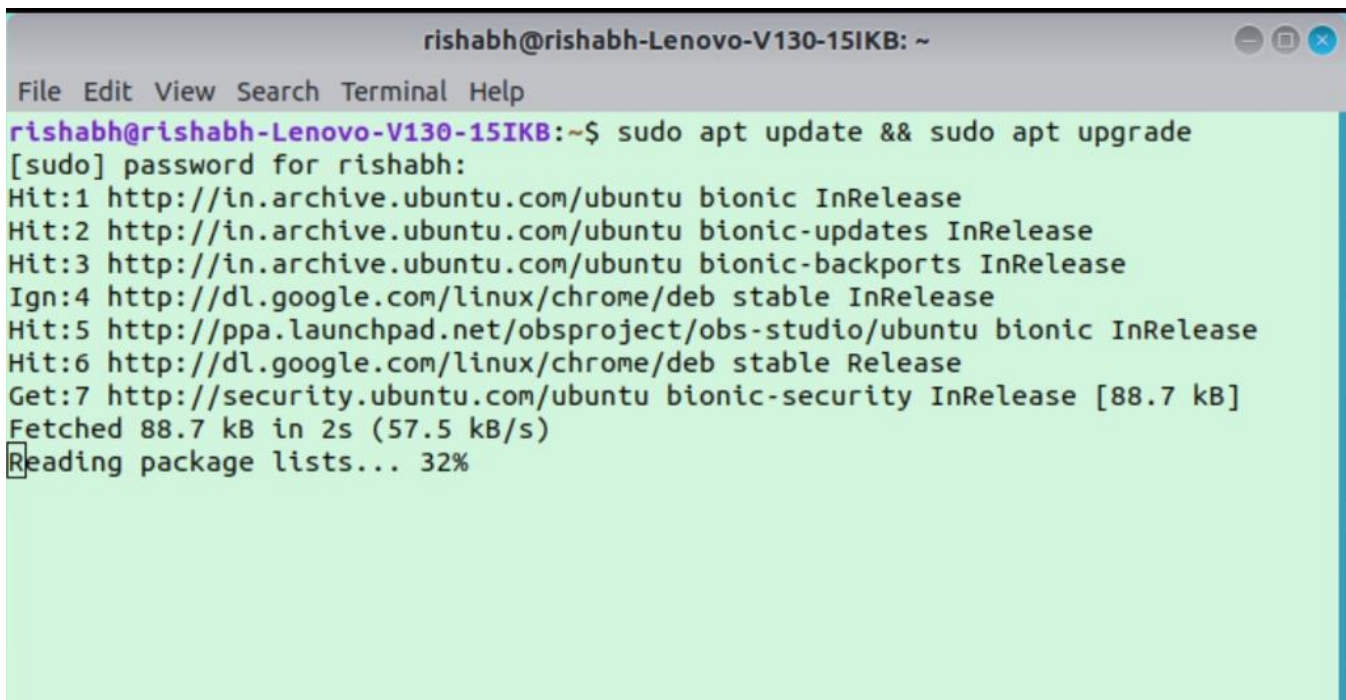
Outcomes: Student will be able to write MongoDB queries for suitable database application using CRUD operations.

Theory:

Installation Process

Step 1: First you need to update and upgrade your system repository in order to install MongoDB. Type the following command in your terminal and then press Enter.

```
$ sudo apt update && sudo apt upgrade
```

A screenshot of a terminal window titled 'rishabh@rishabh-Lenovo-V130-15IKB: ~'. The terminal shows the command 'sudo apt update && sudo apt upgrade' being executed. The output includes several lines of status messages: '[sudo] password for rishabh:', 'Hit:1 http://in.archive.ubuntu.com/ubuntu bionic InRelease', 'Hit:2 http://in.archive.ubuntu.com/ubuntu bionic-updates InRelease', 'Hit:3 http://in.archive.ubuntu.com/ubuntu bionic-backports InRelease', 'Ign:4 http://dl.google.com/linux/chrome/deb stable InRelease', 'Hit:5 http://ppa.launchpad.net/obsproject/obs-studio/ubuntu bionic InRelease', 'Hit:6 http://dl.google.com/linux/chrome/deb stable Release', 'Get:7 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]', 'Fetched 88.7 kB in 2s (57.5 kB/s)', and 'Reading package lists... 32%'. The terminal has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The background of the terminal is light green.

```
rishabh@rishabh-Lenovo-V130-15IKB: ~  
File Edit View Search Terminal Help  
rishabh@rishabh-Lenovo-V130-15IKB:~$ sudo apt update && sudo apt upgrade  
[sudo] password for rishabh:  
Hit:1 http://in.archive.ubuntu.com/ubuntu bionic InRelease  
Hit:2 http://in.archive.ubuntu.com/ubuntu bionic-updates InRelease  
Hit:3 http://in.archive.ubuntu.com/ubuntu bionic-backports InRelease  
Ign:4 http://dl.google.com/linux/chrome/deb stable InRelease  
Hit:5 http://ppa.launchpad.net/obsproject/obs-studio/ubuntu bionic InRelease  
Hit:6 http://dl.google.com/linux/chrome/deb stable Release  
Get:7 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]  
Fetched 88.7 kB in 2s (57.5 kB/s)  
Reading package lists... 32%
```

Step 2: Now, install the MongoDB package using ‘**apt**’. Type the following command and press Enter.

```
$ sudo apt install -y mongodb
```

```
rishabh@rishabh-Lenovo-V130-15IKB: ~  
rishabh@rishabh-Lenovo-V130-15IKB:~$ sudo apt install -y mongodb  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following packages were automatically installed and are no longer required:  
  efibootmgr libfwup1  
Use 'sudo apt autoremove' to remove them.  
The following additional packages will be installed:  
  libboost-program-options1.65.1 libgoogle-perftools4 libpcrecpp0v5  
  libtcmalloc-minimal4 libyaml-cpp0.5v5 mongo-tools mongodb-clients  
  mongodb-server mongodb-server-core  
The following NEW packages will be installed:  
  libboost-program-options1.65.1 libgoogle-perftools4 libpcrecpp0v5  
  libtcmalloc-minimal4 libyaml-cpp0.5v5 mongo-tools mongodb mongodb-clients  
  mongodb-server mongodb-server-core  
0 upgraded, 10 newly installed, 0 to remove and 0 not upgraded.  
Need to get 53.4 MB of archives.  
After this operation, 217 MB of additional disk space will be used.  
Get:1 http://in.archive.ubuntu.com/ubuntu bionic/main amd64 libboost-program-opti  
ons1.65.1 amd64 1.65.1+dfsg-0ubuntu5 [137 kB]  
Get:2 http://in.archive.ubuntu.com/ubuntu bionic/main amd64 libtcmalloc-minimal4  
amd64 2.5-2.2ubuntu3 [91.6 kB]
```

Step 3: Check the service status for MongoDB with the help of following command:

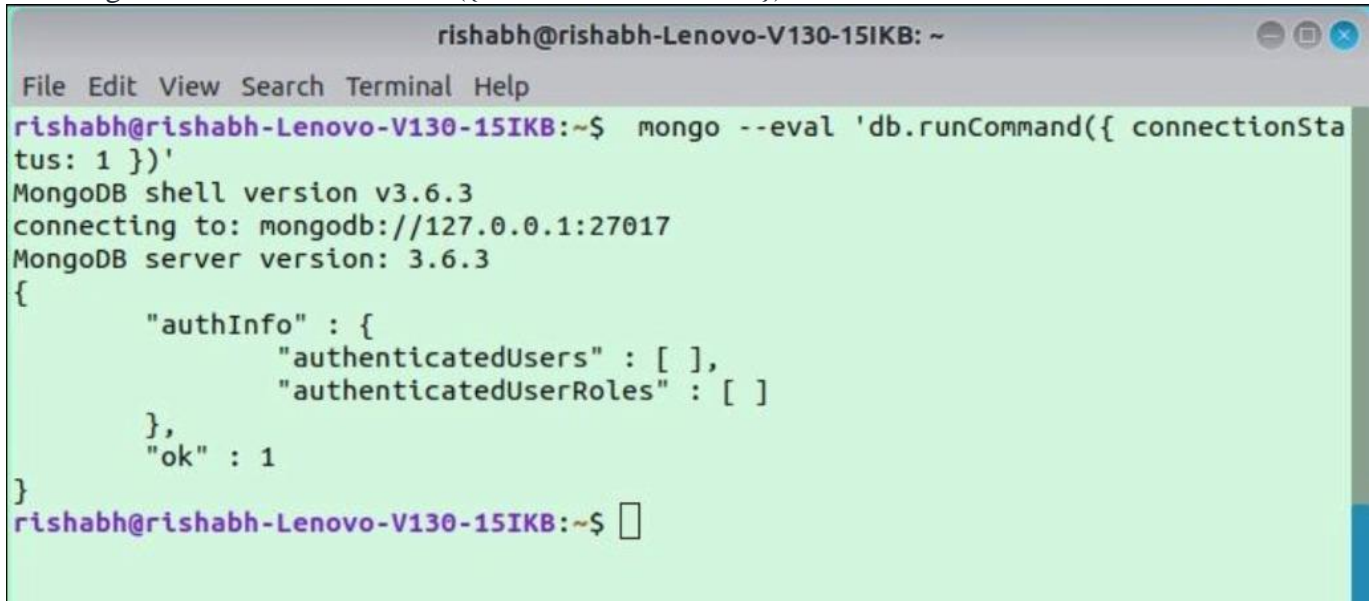
```
rishabh@rishabh-Lenovo-V130-15IKB: ~  
File Edit View Search Terminal Help  
rishabh@rishabh-Lenovo-V130-15IKB:~$ sudo systemctl status mongodb  
● mongodb.service - An object/document-oriented database  
   Loaded: loaded (/lib/systemd/system/mongodb.service; enabled; vendor preset: e  
   Active: active (running) since Tue 2020-03-03 15:49:56 IST; 29s ago  
     Docs: man:mongod(1)  
  Main PID: 14151 (mongod)  
    Tasks: 23 (limit: 4915)  
   CGroup: /system.slice/mongodb.service  
           └─14151 /usr/bin/mongod --unixSocketPrefix=/run/mongodb --config /etc/  
  
Mar 03 15:49:56 rishabh-Lenovo-V130-15IKB systemd[1]: Started An object/document-  
lines 1-10/10 (END)
```

```
$ sudo systemctl status mongodb
```

systemctl verifies that MongoDB server is up and running.

Step 4: Now check if the installation process is done correctly and everything is working fine. Go through the following command:

```
$ mongo --eval 'db.runCommand({ connectionStatus: 1 })'
```



```
rishabh@rishabh-Lenovo-V130-15IKB: ~  
File Edit View Search Terminal Help  
rishabh@rishabh-Lenovo-V130-15IKB:~$ mongo --eval 'db.runCommand({ connectionSta  
tus: 1 })'  
MongoDB shell version v3.6.3  
connecting to: mongodb://127.0.0.1:27017  
MongoDB server version: 3.6.3  
{  
  "authInfo" : {  
    "authenticatedUsers" : [ ],  
    "authenticatedUserRoles" : [ ]  
  },  
  "ok" : 1  
}  
rishabh@rishabh-Lenovo-V130-15IKB:~$
```

the value “1” in ok field indicates that the server is working properly with no errors.

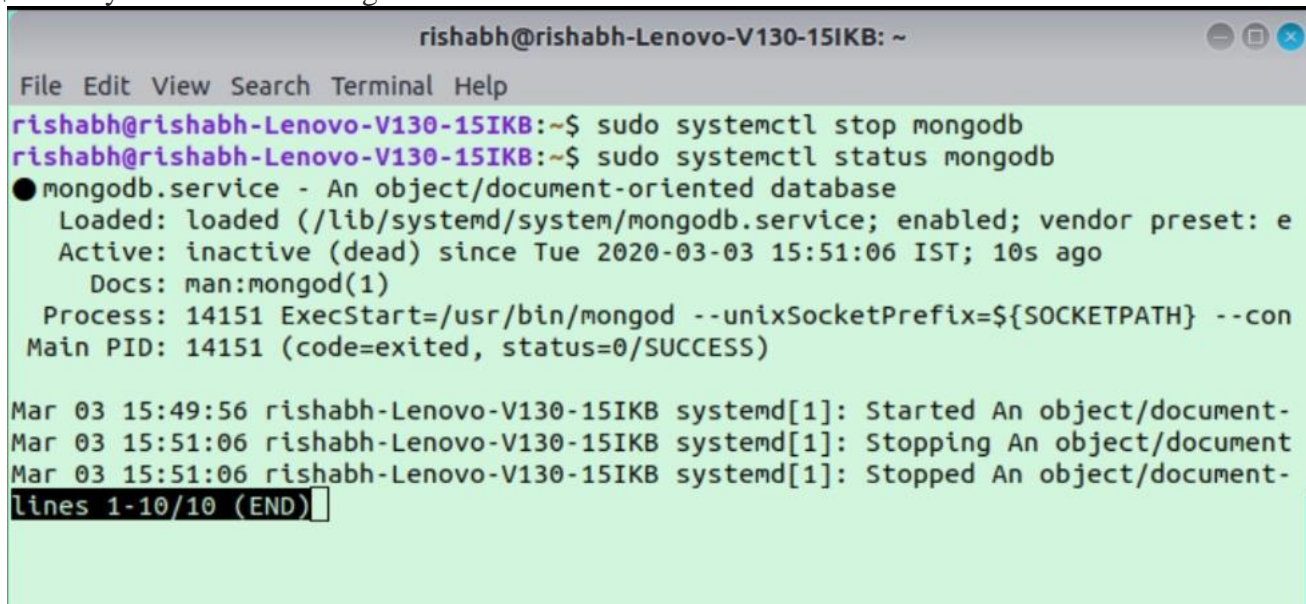
Step 5: MongoDB services can be started and stopped with the use of following commands:

To stop running the MongoDB service, use command :

```
$ sudo systemctl stop mongod
```

MongoDB service has been stopped and can be checked by using the status command:

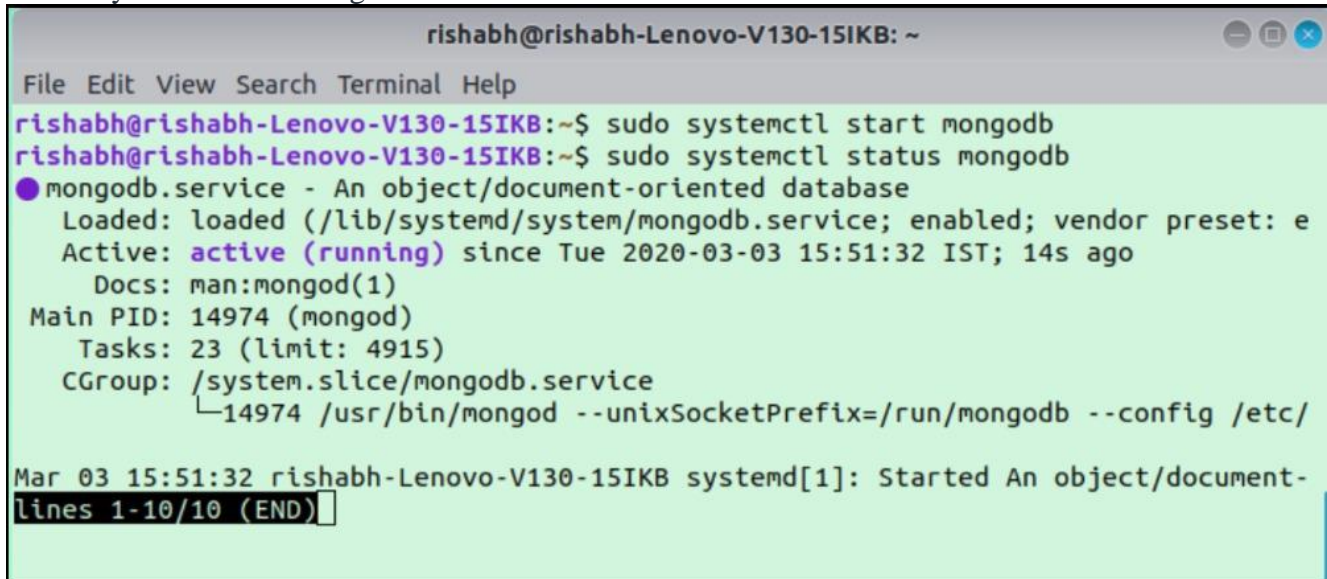
```
$ sudo systemctl status mongod
```



```
rishabh@rishabh-Lenovo-V130-15IKB: ~  
File Edit View Search Terminal Help  
rishabh@rishabh-Lenovo-V130-15IKB:~$ sudo systemctl stop mongod  
rishabh@rishabh-Lenovo-V130-15IKB:~$ sudo systemctl status mongod  
● mongod.service - An object/document-oriented database  
   Loaded: loaded (/lib/systemd/system/mongod.service; enabled; vendor preset: e  
   Active: inactive (dead) since Tue 2020-03-03 15:51:06 IST; 10s ago  
     Docs: man:mongod(1)  
   Process: 14151 ExecStart=/usr/bin/mongod --unixSocketPrefix=${SOCKETPATH} --con  
   Main PID: 14151 (code=exited, status=0/SUCCESS)  
  
Mar 03 15:49:56 rishabh-Lenovo-V130-15IKB systemd[1]: Started An object/document-  
Mar 03 15:51:06 rishabh-Lenovo-V130-15IKB systemd[1]: Stopping An object/document  
Mar 03 15:51:06 rishabh-Lenovo-V130-15IKB systemd[1]: Stopped An object/document-  
lines 1-10/10 (END)
```


As it can be seen that the service has stopped, to start the service we can use :

\$ sudo systemctl start mongod

A terminal window titled 'rishabh@rishabh-Lenovo-V130-15IKB: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output:

```
rishabh@rishabh-Lenovo-V130-15IKB:~$ sudo systemctl start mongod
rishabh@rishabh-Lenovo-V130-15IKB:~$ sudo systemctl status mongod
● mongod.service - An object/document-oriented database
   Loaded: loaded (/lib/systemd/system/mongod.service; enabled; vendor preset: e
   Active: active (running) since Tue 2020-03-03 15:51:32 IST; 14s ago
     Docs: man:mongod(1)
  Main PID: 14974 (mongod)
    Tasks: 23 (limit: 4915)
   CGroup: /system.slice/mongod.service
           └─14974 /usr/bin/mongod --unixSocketPrefix=/run/mongod --config /etc/

Mar 03 15:51:32 rishabh-Lenovo-V130-15IKB systemd[1]: Started An object/document-
lines 1-10/10 (END)
```

\$mongod :for run mongod on terminal.

Crud operation In mongodb CRUD:-

create,read,update,delete.

Create:

db.collection.insertOne()	It is used to insert a single document in the collection.
db.collection.insertMany()	It is used to insert multiple documents in the collection.
db.createCollection()	It is used to create an empty collection.

Create database

>use database_name“eg:use
aids”

>db.tablename.insertOne({name:”Aaditya”,age:20,branch:”AIDS”}) Or
db.tablename.insert({name:”Aaditya”,age:20,branch:”AIDS”})

Run successful: { “acknowledged:true,”
“insertedId”:ObjectId(“...”)
}

```
>db.tablename.insertMany([ {name:"Aaditya",age:20,branch:"AIDS"}, {name:"Anurag",age=21,branch="AIDS"}, {name:"Vaibhav",age=20,branch:"E&TC"} ])
```

Run successful: { "acknowledged:true,"

"insertedId":

[ObjectId("..."),

ObjectId("...")

ObjectId("...")]}

Read:

db.collection.find() It is used to retrieve documents from the collection

```
>db.tablename.find().pretty()
```

```
>db.tablename.find()
```

Update:

db.collection.updateOne()

It is used to update a single document in the collection that satisfy the given criteria.

db.collection.updateMany()

It is used to update multiple documents in the collection that satisfy the given criteria.

db.collection.replaceOne()

It is used to replace single document in the collection that satisfy the given criteria.

```
> db.table_name.updateOne({name:"Aaditya"},{$set:{age=21,name="adi"}})
```

```
Or db.table_name.update({name:"Aaditya"},{$set:{age=21,name="adi"}})
```

```
>db.table_name.updateMany({},{$set:{year:2020 }})Add year column
```

```
> db.empDetails.findOneAndUpdate({},{$set:{}})It Used for update and show table
```

Delete:

db.collection.deleteOne()

It is used to delete a single document from the collection that satisfy the given criteria.

db.collection.deleteMany()

It is used to delete multiple documents from the collection that satisfy the given criteria.

```
>db.table_name.deleteOne({name:"Anurag"})
```

```
>db.table_name.deleteMany({})
```

remove():

```
db.table_name.remove({name:"Anurag"})
```

Save Method:

The db.collection.save() method is used to updates an existing document or inserts a new document, depending on its document parameter

```
>db.table_name.save()
```

Logical Operator:

\$and

It is used to join query clauses with a logical AND and return all documents that match the given conditions of both clauses.

\$or

It is used to join query clauses with a logical OR and return all documents that match the given conditions of either clause.

\$not

It is used to invert the effect of the query expressions and return documents that does not match the query expression.

\$nor

It is used to join query clauses with a logical NOR and return all documents that fail to match both clauses.

And:

```
>db.table_name.find({$and: [{branch: "AIDS"}, {joiningYear: 2022}]}).pretty()Or:
```

```
>db.contributor.find({$or: [{branch: "AIDS"}, {joiningYear: 2022}]}).pretty()Not:
```

In this example, we are retrieving only those employee's documents whose salary is not greater than 2000

```
>db.table_name.find({salary: {$not: {$gt: 2000}}}).pretty()Nor:
```

In this example, we are retrieving only those employee's documents whose salary is not 3000 and whose branch is not ECE

```
>db.table_name.find({$nor: [{salary: 3000}, {branch: "ECE"}]}).pretty()
```

Assignment No. 4

Title: Unnamed PL/SQLcode block: Use of Control structure and Exception handling is mandatory.

Suggested Problem statement:

2. Consider Tables:
 3. Borrower(Roll_no, Name, Date of Issue, Name of Book, Status)
 4. Fine(Roll_no, Date, Amt)
 - G. Accept Roll_no and Name of Book from the user.
 - H. Check the number of days (from date of issue).
 - I. If days are between 15 to 30 then the fine amount will be Rs 5per day.
 - J. If no. of days>30, per day fine will be Rs 50 per day and for days less than 30, Rs. 5 per day.
 - K. After submitting the book, status will change from I to R.
 - L. if condition of fine is true, then details will be stored into fine table
- Also handles the exception by named exception handler or user defined exception handler

Theory:

mysql> use Abhi;

Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed

mysql> delimiter //

mysql> call B1(1,'TOC') //

```
+.....+
| NOT FOUND |
+-----+| NOT FOUND |
+.....+
1 row in set (0.35 sec)
Query OK, 0 rows affected (0.41 sec)
```

mysql> select * from Borrower;

```
-> //
+.....+.....+.....+.....+.....+
| roll_no | name
| DOI
| book_name | status
|
+.....+.....+.....+.....+.....+
| 12 | patel | 2018-07-01 | xyz | issued |
| 14 | shinde | 2018-06-01 | oop | issued |
| 16 | bhangale | 2018-05-01 | coa | returned |
| 18 | rebello | 2018-06-15 | toc | returned |
| 20 | patil | 2018-05-15 | mp | issued
|
+.....+.....+.....+.....+.....+
5 rows in set (0.00 sec)
```

```

mysql> show tables;
-> //
+-----+
| Tables_in_Abhi |
+-----+
| Borrower |
| Employee |
| Fine |
| TE |
| _master |
| auto |
| c_master |
| capital || customer |
| orders |
| person |
| product_master |
| state |
|
+-----+
13 rows in set (0.00 sec)
mysql> create procedure B(roll_new int,book_name varchar(20))
-> begin
-> declare X integer;
-> declare continue handler for not found
-> begin
-> select 'NOT FOUND';
-> end;
-> select datediff(curdate(),DOI) into X from Borrower
where roll_no=roll_new;
->
if (X>15&&X<30)
-> then
-> insert into Fine values(roll_new,curdate(),(X*5));
-> end if;
-> if (X>30)
-> then
-> insert into Fine values(roll_new,curdate(),(X*50));
-> end if;
-> update Borrower set status='returned' where
roll_no=roll_new;
-> end;
-> //
Query OK, 0 rows affected (0.02 sec)
mysql> call B(12,'xyz');-> //
Query OK, 1 row affected (0.42 sec)
mysql> select * from Fine;
+-----+-----+-----+

```

```

| roll_no | fine_date
| amount |
+-----+-----+-----+
|
12 | 2018-07-28 |
135 |
+-----+-----+-----+
1 row in set (0.00 sec)
mysql> select * from Borrower;
+-----+-----+-----+-----+-----+
| roll_no | name
| DOI
| book_name | status
|
+-----+-----+-----+-----+-----+
| 12 | patel | 2018-07-01 | xyz | returned |
| 14 | shinde | 2018-06-01 | oop | issued
| 16 | bhangale | 2018-05-01 | coa | returned |
| 18 | rebello | 2018-06-15 | toc | returned |
| 20 | patil | 2018-05-15 | mp | issued
||
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
mysql> call B(20,'patil');
-> //
Query OK, 1 row affected (0.35 sec)
mysql> select * from Fine;
+-----+-----+
| roll_no | fine_date
| amount |
+-----+-----+
|
12 | 2018-07-28 |
135 ||
20 | 2018-07-28 |
3700 |
+-----+-----+
2 rows in set (0.00 sec)
mysql> select * from Borrower;
+-----+-----+-----+-----+-----+
| roll_no | name
| DOI
| book_name | status
|
+-----+-----+-----+-----+-----+
| 12 | patel | 2018-07-01 | xyz | returned |
| 14 | shinde | 2018-06-01 | oop | issued

```

```
| 16 | bhangale | 2018-05-01 | coa | returned |
| 18 | rebello | 2018-06-15 | toc | returned |
| 20 | patil | 2018-05-15 | mp | returned |
|
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
mysql>
```

Assignment No. 5

Title: Exporting and Importing data

- Design and develop SQL DML statements to demonstrate **exporting** tables to external files of different file formats ex. CSV, XLSX, TXT, etc.
- Design and develop SQL DML statements to demonstrate **importing** data from external files of different file formats ex. CSV, XLSX, TXT, etc.

Theory:

Exporting Data:

- Backups are important!
- Backup principles:

- Perform regularly.
- Provide consistent and meaningful naming scheme.
- Expire backup files.
- Include with regular system backups.

- Reasons for exporting data:

Copying databases from one server to another:

- From within the same host
- To another host
- Testing a new MySQL release with real data
- Transferring data from one RDBMS to another

Exporting with a Query:

- Write query results directly into a file by using SELECT with the INTO OUTFILE clause.
- Example:

```
SELECT * INTO OUTFILE 'D:/DataBackup/Country.txt'  
FROM Country
```

- The name of the file indicates the location of the writ

Default file format:

- Values are delimited by tab characters.
- Lines are terminated with newlines.
- The file format can be changed by using specific INTO OUTFILE options.

Exporting with a Query: INTO OUTFILE

INTO OUTFILE changes the standard SELECT operation:

- A file is written to the server host rather than to the executing client.
- Data is written to a new file only; there are no overwrites.
- The file contains one line per row selected by the statement.
- The user that executes the statement must have the FILE privilege.

- The file is created with file system access permissions.

Exporting with a Query: CSV Format

- Create a text file that contains information in comma-separated values (CSV) format:
- Values are enclosed in double quotation marks.
- Lines are terminated by carriage returns.

- Example:

```
SELECT * INTO OUTFILE '/tmp/table1.txt'  
FIELDS TERMINATED BY ',' ENCLOSED BY '"'  
LINES TERMINATED BY '\r'  
FROM table1
```

Importing Data

New data can be imported into a database by using a MySQL (.sql) statement file, which contains all the information needed to create tables.

- Example: Import the world_innodb database from file by using the mysql command.

```
mysql -uusername -ppassword world_innodb <  
world_innodb.sql
```

Use the input operator (<) to indicate the SQL file name.

- Example: Import the data file within the mysql client

```
SOURCE D:/DataBackup/world_innodb.sql
```

Caution: Do not import over an existing database file with the same name.
Shell-level client commands do not require a semicolon (;) at the end of the statement.
A text file (.txt) can also be used to import data.

Importing from a Data File:

You should know the following characteristics of the data file:

- The column value separator
- The order of the columns
- The row separator
- The file system where the file resides
- What are the values enclosed within? (example: double quotation marks)
- Are the column names specified in the file?
- Is there a header indicating rows of the table to skip before importing?
- Are privileges required to access the file?

Importing with the LOAD DATA INFILE Statement

- LOAD DATA INFILE is the reverse operation of SELECT with INTO OUTFILE.
- However, it uses similar clauses and format specifiers.
- It reads row values from a file into a table.
- Files can be in tab-delimited or comma-separated format.
- Example:

```
LOAD DATA INFILE 'D:/DataBackup/City.txt'  
INTO TABLE city
```

MySQL assumes that the file is located on the server host in the database data directory.

The simplest form of the LOAD DATA INFILE statement specifies only the name of the data file and the table into which to load the file, as shown in the example in the slide.

The syntax for LOAD DATA INFILE is as follows, where optional parts of the statement are indicated by square brackets:

LOAD DATA [LOCAL] INFILE 'file_name'

[IGNORE | REPLACE]

INTO TABLE table_name

format_specifiers

[IGNORE n LINES]

[(column_list)]

[SET (assignment_list)]

Importing with LOAD DATA INFILE: CSV Format

- Import a text file that contains information in comma-separated values format.
- Example:

```
LOAD DATA INFILE '/tmp/table1.txt'  
INTO TABLE table1  
FIELDS TERMINATED BY ',' ENCLOSED BY '"'  
LINES TERMINATED BY '\r'
```

The file name is given as a quoted string. On Windows, the pathname separator character is “\”, but MySQL treats the backslash as the escape character in strings. To deal with this issue, write separators in Windows pathnames either as “/” or as “\\”. To load a file named D:\mydata\data.txt, specify the file name as shown in either of the following statements:

LOAD DATA INFILE 'D:\\mydata\\data.txt' INTO TABLE t

The example in the slide shows how to use a **LOAD DATA INFILE** statement to import a file named **/tmp/data.txt** that contains information in comma-separated values (CSV) format, with values enclosed within double quotation marks and lines terminated by carriage returns into a table named **table1**.

Assignment No. 6

Title: Database Connectivity:

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

Theory:

MySQL and Java JDBC. This tutorial describes how to use Java JDBC to connect to MySQL and perform SQL queries, database inserts and deletes.

1. Connection to database with Java

The interface for accessing relational databases from Java is *Java Database Connectivity (JDBC)*. Via JDBC you create a connection to the database, issue database queries and update as well as receive the results.

JDBC provides an interface which allows you to perform SQL operations independently of the instance of the used database. To use JDBC, you require the database specific implementation of the JDBC driver.

2. Introduction to MySQL

To learn to install and use MySQL please see [MySQL - Tutorial](#).

The following description will assume that you have successfully installed MySQL and know how to access MySQL via the command line.

3. MySQL JDBC driver

To connect to MySQL from Java, you have to use the JDBC driver from MySQL. The MySQL JDBC driver is called *MySQL Connector/J*. You find the latest MySQL JDBC driver under the following URL: <http://dev.mysql.com/downloads/connector/j>.

The download contains a **JAR** file which we require later.

4. Exercise: create example database

In this exercise you create a new database, a new user and an example table. For this connect to the MySQL server via the **mysql** command line client.

Create a new database called *feedback* and start using it with the following command.

```
create database feedback;  
use feedback;
```

Create a user with the following command.

```
CREATE USER sqluser IDENTIFIED BY 'sqluserpw';
```

```
grant usage on *.* to sqluser@localhost identified by 'sqluserpw';  
grant all privileges on feedback.* to sqluser@localhost;
```

Now create a sample database table with example content via the following SQL statement.

```
CREATE TABLE comments (  
    id INT NOT NULL AUTO_INCREMENT,  
    MYUSER VARCHAR(30) NOT NULL,  
    EMAIL VARCHAR(30),  
    WEBPAGE VARCHAR(100) NOT NULL,  
    DATUM DATE NOT NULL,  
    SUMMARY VARCHAR(40) NOT NULL,  
    COMMENTS VARCHAR(400) NOT NULL,  
    PRIMARY KEY (ID)  
);
```

```
INSERT INTO comments values (default, 'lars',  
'myemail@gmail.com','https://www.vogella.com/', '2009-09-14 10:33:11', 'Summary','My first  
comment' );
```

5. Java JDBC

Create a Java project and a package called *de.vogella.mysql.first*.

Create a `lib` folder and copy the JDBC driver into this folder. Add the JDBC driver to your classpath. See [Adding jars to the classpath](#) for details.

Create the following class to connect to the MySQL database and perform queries, inserts and deletes. It also prints the metadata (table name, column names) of a query result.

```
package de.vogella.mysql.first;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
import java.util.Date;
```

```

public class MySQLAccess {
    private Connection connect = null;
    private Statement statement = null;
    private PreparedStatement preparedStatement = null;
    private ResultSet resultSet = null;

    public void readDataBase() throws Exception {
        try {
            // This will load the MySQL driver, each DB has its own driver
            Class.forName("com.mysql.jdbc.Driver");
            // Setup the connection with the DB
            connect = DriverManager
                .getConnection("jdbc:mysql://localhost/feedback?"
                    + "user=sqluser&password=sqluserpw");

            // Statements allow to issue SQL queries to the database
            statement = connect.createStatement();
            // Result set get the result of the SQL query
            resultSet = statement
                .executeQuery("select * from feedback.comments");
            writeResultSet(resultSet);

            // PreparedStatements can use variables and are more efficient
            preparedStatement = connect
                .prepareStatement("insert into feedback.comments values (default, ?, ?, ?, ? , ?,
?)");
            // "myuser, webpage, datum, summary, COMMENTS from feedback.comments");
            // Parameters start with 1
            preparedStatement.setString(1, "Test");
            preparedStatement.setString(2, "TestEmail");
            preparedStatement.setString(3, "TestWebpage");
            preparedStatement.setDate(4, new java.sql.Date(2009, 12, 11));
            preparedStatement.setString(5, "TestSummary");
            preparedStatement.setString(6, "TestComment");
            preparedStatement.executeUpdate();

            preparedStatement = connect
                .prepareStatement("SELECT myuser, webpage, datum, summary,
COMMENTS from feedback.comments");
            resultSet = preparedStatement.executeQuery();
            writeResultSet(resultSet);

            // Remove again the insert comment
            preparedStatement = connect
                .prepareStatement("delete from feedback.comments where myuser= ? ; ");
            preparedStatement.setString(1, "Test");

```

```
preparedStatement.executeUpdate();
resultSet = statement
.executeQuery("select * from feedback.comments");
writeMetaData(resultSet);
} catch (Exception e) {
throw e;
} finally {
close();
}
} private void writeMetaData(ResultSet resultSet) throws SQLException {
// Now get some metadata from the database
// Result set get the result of the SQL query System.out.println("The columns in the table are: ");
System.out.println("Table: " + resultSet.getMetaData().getTableName(1));
for (int i = 1; i<= resultSet.getMetaData().getColumnCount(); i++)
{ System.out.println("Column " +i + " "+ resultSet.getMetaData().getColumnName(i));
}
}
private void writeResultSet(ResultSet resultSet) throws SQLException {
// ResultSet is initially before the first data set
while (resultSet.next()) {
// It is possible to get the columns via name
// also possible to get the columns via the column number
// which starts at 1
// e.g. resultSet.getSTring(2);
String user = resultSet.getString("myuser");
String website = resultSet.getString("webpage");
String summary = resultSet.getString("summary");
Date date = resultSet.getDate("datum");
String comment = resultSet.getString("comments");
System.out.println("User: " + user);
System.out.println("Website: " + website);
System.out.println("summary: " + summary);
System.out.println("Date: " + date);
System.out.println("Comment: " + comment);
}
}
// You need to close the resultSet
```

```
private void close() {  
try {  
    if (resultSet != null) {  
        resultSet.close();  
    }  
    if (statement != null) { statement.close();  
    } if (connect != null) {  
        connect.close();  
    }  
    } catch (Exception e) {  
    }  
    }  
}
```

Create the following main program to test your class.

```
package de.vogella.mysql.first.test;  
  
import de.vogella.mysql.first.MySQLAccess;  
  
public class Main {  
    public static void main(String[] args)  
        throws Exception { MySQLAccess dao = new MySQLAccess();  
        dao.readDataBase();  
    }  
}
```

Assignment No. 7

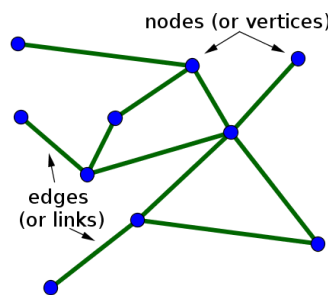
Title: Implement depth first search algorithm and Breadth First Search algorithm. Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure.

Theory:

Undirected Graph:

An undirected graph is graph, i.e., a set of objects (called vertices or nodes) that are connected together, where all the edges are bidirectional. An undirected graph is sometimes called an undirected network. In contrast, a graph where the edges point in a direction is called a directed graph.

When drawing an undirected graph, the edges are typically drawn as lines between pairs of nodes, as illustrated in the following figure.



Depth First Search:

Depth-first search is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking.

So the basic idea is to start from the root or any arbitrary node and mark the node and move to the adjacent unmarked node and continue this loop until there is no unmarked adjacent node. Then backtrack and check for other unmarked nodes and traverse them. Finally, print the nodes in the path. DFS uses a **stack data structure** for traversal.

Time Complexity: $O(V+E)$, where V is the number of nodes and E is the number of edges.

Space Complexity: $O(V)$

Breadth First Search:

Breadth-First Traversal (or Search) for a graph is similar to Breadth-First Traversal of a tree.

The only catch here is, that, unlike trees, graphs may contain cycles, so we may come to the same node again. To avoid processing a node more than once, we divide the vertices into two categories:

- Visited and
- Not visited.

A boolean visited array is used to mark the visited vertices. For simplicity, it is assumed that all vertices are reachable from the starting vertex. BFS uses a **queue data structure** for traversal.

Time Complexity: $O(V+E)$, where V is the number of nodes and E is the number of edges.

Space Complexity: $O(V)$

Program: }

```
from collections import defaultdict, deque

# Function to add an edge to the graph
def add_edge(graph, u, v):
    graph[u].append(v)
    graph[v].append(u)

# Depth-First Search (DFS) function
def dfs(graph, node, visited):
    if node not in visited:
        print(node, end=' ') # Print the current node
        visited.add(node)    # Mark the node as visited
        for neighbor in graph[node]:
            dfs(graph, neighbor, visited)

# Breadth-First Search (BFS) function
def bfs(graph, start):
    visited = set()
    queue = deque([start])

    while queue:
        node = queue.popleft()
        if node not in visited:
            print(node, end=' ') # Print the current node
            visited.add(node)
            queue.extend(neighbor for neighbor in graph[node] if neighbor not in
visited)

# Get the number of vertices and edges from the user
num_vertices = int(input("Enter the number of vertices: "))
num_edges = int(input("Enter the number of edges: "))

# Create an empty graph as an adjacency list
graph = defaultdict(list)

# Input edges
for _ in range(num_edges):
    u, v = input("Enter an edge (format: u v): ").split()
    add_edge(graph, u, v)

# Choose a starting node for DFS and BFS
start_node = input("Enter the starting node: ")

# Initialize visited sets
```



```
visited_dfs = set()
visited_bfs = set()

# Perform DFS and BFS
print("Depth-First Search (DFS):")
dfs(graph, start_node, visited_dfs)
print("\n")

print("Breadth-First Search (BFS):")
bfs(graph, start_node)
```

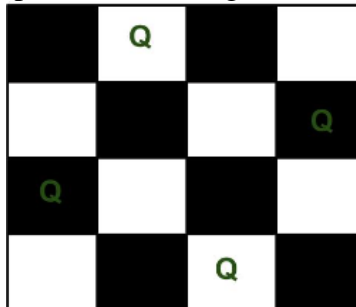
Output:**Enter the number of vertices: 6****Enter the number of edges: 7****Enter an edge (format: u v): A B****Enter an edge (format: u v): A C****Enter an edge (format: u v): B D****Enter an edge (format: u v): B E****Enter an edge (format: u v): C F****Enter an edge (format: u v): D E****Enter an edge (format: u v): F E****Enter the starting node: A****Depth-First Search (DFS):****A B D E F C****Breadth-First Search (BFS):****A B C D E F**

Assignment No. 8

Title: Implement n-queens problem.

Theory:

The N Queen is the problem of placing N chess queens on an N×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 Method

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.

Algorithm:

- 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 works, return false to trigger backtracking.

Program:

```

def is_safe(board, row, col):
    """
    Check if it's safe to place a queen at a specific position (row, col) on the
    board.
    """
    # Check the same column above this row
    for i in range(row):
        if board[i][col] == 1:
            return False

    # Check upper-left diagonal
    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    # Check upper-right diagonal
    for i, j in zip(range(row, -1, -1), range(col, len(board))):
        if board[i][j] == 1:
            return False

    return True

def solve_n_queens(N):
    """
    Solve the N-Queens problem for a given board size (N x N).
    """
    board = [[0] * N for _ in range(N)] # Create an empty chessboard
    solutions = [] # List to store all solutions

    def backtrack(row):
        # If we've placed queens in all rows, we've found a solution
        if row == N:
            solutions.append(["".join("Q" if cell == 1 else "." for cell in row)
                               for row in board])
            return

        # Try placing a queen in each column of the current row
        for col in range(N):
            if is_safe(board, row, col):
                board[row][col] = 1 # Place the queen
                backtrack(row + 1) # Recur to the next row
                board[row][col] = 0 # Backtrack (remove the queen)

    backtrack(0) # Start with the first row

```

```
        return solutions

def print_solutions(solutions):
    """
    Print all solutions to the N-Queens problem.
    """
    for i, solution in enumerate(solutions):
        print(f"Solution {i + 1}:")
        for row in solution:
            print(row)
        print()

# Example usage with N = 4 (4-Queens problem)
N = 4
solutions = solve_n_queens(N)

if solutions:
    print_solutions(solutions)
else:
    print("No solution found.")
```

Output:**Solution 1:**

```
0 1 0 0
0 0 0 1
1 0 0 0
0 0 1 0
```

Solution 2:

```
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
```

Assignment No. 9

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

Theory:

Alpha-Beta Tree Search, also known as 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.

Program:

MAX = 1000

MIN = -1000

```
def minimax(depth, nodeIndex, maximizingPlayer, values, alpha, beta):
    if depth == 3:
        return values[nodeIndex]

    if maximizingPlayer:
        best = MIN
        for i in range(2):
            val = minimax(depth + 1, nodeIndex * 2 + i, False, values, alpha,
beta)
            best = max(best, val)
            alpha = max(alpha, best)
            if beta <= alpha:
                break
        return best
    else:
        best = MAX
        for i in range(2):
            val = minimax(depth + 1, nodeIndex * 2 + i, True, values, alpha,
beta)
            best = min(best, val)
            beta = min(beta, best)
```

```
        if beta <= alpha:
            break
    return best

if __name__ == "__main__":
    values = [3, 5, 6, 9, 1, 2, 0, -1]
    print("The optimal value is:", minimax(0, 0, True, values, MIN, MAX))
```

Output:

The optimal value is : 5

Assignment No. 10

Title: Implement Greedy search algorithm for Selection Sort

Theory:

The **selection sort algorithm** sorts an array by repeatedly finding the minimum element (considering ascending order) from the unsorted part and putting it at the beginning.

The algorithm maintains two subarrays in a given array.

- The subarray which already sorted.
- The remaining subarray was unsorted.

In every iteration of the selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.

Follow the below steps to solve the problem:

- Initialize minimum value(**min_idx**) to location 0.
- Traverse the array to find the minimum element in the array.
- While traversing if any element smaller than **min_idx** is found then swap both the values.
- Then, increment **min_idx** to point to the next element.
- Repeat until the array is sorted.

Time Complexity: The time complexity of Selection Sort is $O(N^2)$ as there are two nested loops:

- One loop to select an element of Array one by one = $O(N)$
- Another loop to compare that element with every other Array element = $O(N)$

Therefore overall complexity = $O(N) * O(N) = O(N*N) = O(N^2)$

Space Complexity: $O(1)$ as the only extra memory used is for temporary variables while swapping two values in Array. The selection sort never makes more than $O(N)$ swaps and can be useful when memory write is a costly operation.

Program:

```
def swap(xp, yp):  
    temp = xp  
    xp = yp  
    yp = temp  
  
def selectionSort(arr):  
    n = len(arr)  
    for i in range(n - 1):  
        min_idx = i  
        for j in range(i + 1, n):
```

```
        if arr[j] <
            arr[min_idx]:min_idx
            = j
    if min_idx != i:
        swap(arr[min_idx], arr[i])

def
    printArray(arr)
    :for i in arr:
        print(i, end="
    ")print()

if __name__ == "__main__":
    arr = [64, 25, 12, 22,
        11]
```

Output:

Sorted array:

11 12 22 25 64

Assignment No. 11

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

Theory:

Chatbots are the computer program that uses Artificial intelligence system that interacts with the user through text or voice. It saves time with automation.

It is a simple project in C++ that uses an Artificial Intelligence system that can simulate a conversation with a user through text and give answers through text or voice. The main aim of this project is to develop a chatbot which is works in online and offline mode.

Program:

```
import os
import time
import subprocess

def wish_me():
    current_time = time.localtime()
    if current_time.tm_hour < 12:
        print("Good Morning Sir")
        os.system("espeak 'Good Morning Sir'")
    elif 12 <= current_time.tm_hour < 17:
        print("Good Afternoon Sir")
        os.system("espeak 'Good Afternoon Sir'")
    else:
        print("Good Evening Sir")
        os.system("espeak 'Good Evening Sir'")

def get_date_time():
    current_time = time.ctime()
    print("The date and time is:")
    print(current_time)

if __name__ == "__main__":
    print("\t\t\t<===== W E L C O M E
=====>")
    print("\t\t\t<===== I'M A VIRTUAL ASSISTANT
=====>")

    password = input("Enter your password: ")

    while password != "chauhan":
```

```

print("\n<=====
=====>\n")
    print("\t\t<===== W E L C O M E
=====>")
    print("\t\t<===== I'M VIRTUAL ASSISTANT
=====>")
    print("\n=====")
    print("| Incorrect Password |")
    print("=====")
    password = input("\nEnter your password: ")

print("\n<=====
=====>\n")
    wish_me()

    while True:

print("\n<=====
=====>\n")
    user_input = input("\nHow can I help you, sir: ")

    if user_input.lower() in ["hi", "hey", "hello"]:
        print("Hello Sir...")
        os.system("espeak 'Hello Sir'")
    elif user_input.lower() in ["bye", "stop", "exit"]:
        print("Goodbye, sir. Have a nice day!")
        os.system("espeak 'Goodbye, sir. Have a nice day'")
        break
    elif user_input.lower() in ["who are you", "tell me about yourself",
"about"]:
        print("I'm a virtual assistant created by Aditi!")
        os.system("espeak 'I am a virtual assistant created by Aditi'")
    elif user_input.lower() in ["how are you", "whatsup", "how is your day"]:
        print("I'm good, sir. Tell me, how can I help you?")
        os.system("espeak 'I'm good, sir. Tell me, how can I help you'")
    elif user_input.lower() in ["time", "date"]:
        get_date_time()
    elif user_input.lower() == "open notepad":
        print("Opening Notepad...")
        os.system("espeak 'Opening Notepad'")
        subprocess.Popen(["notepad.exe"])
    elif user_input.lower() == "open google":
        print("Opening Google...")

```

```

        os.system("espeak 'Opening Google'")
        subprocess.Popen(["start", "https://www.google.com"])
    elif user_input.lower() == "open youtube":
        print("Opening YouTube...")
        os.system("espeak 'Opening YouTube'")
        subprocess.Popen(["start", "https://www.youtube.com"])
    elif user_input.lower() == "open instagram":
        print("Opening Instagram...")
        os.system("espeak 'Opening Instagram'")
        subprocess.Popen(["start", "https://www.instagram.com"])
    else:
        print("Sorry, I could not understand your query. Please try again!")
        os.system("espeak 'Sorry, I could not understand your query. Please
try again'")

```

Output

<===== W E L C O M E

=====>

<===== I'M A VIRTUAL ASSISTANT

=====>

Enter your password: chauhan

<=====

=====>

Good Afternoon Sir

<=====

=====>

How can I help you, sir: hi

Hello Sir...

<=====

=====>

How can I help you, sir: time

The date and time is:

Sun Oct 9 15:30:00 2023

<=====

=====>

How can I help you, sir: open notepad

Opening Notepad...

<=====

=====>

How can I help you, sir: bye

Goodbye, sir. Have a nice day!

