

Class XII ICT Practical

Ashish Kumar Bhoi, P.G.T. Computer Science
Odisha Adarsha Vidyalaya Bibina, Saintala
ashish@akbhoi.com

April 1, 2026

Contents

1	Creation of a Database	7
1.1	Aim	7
1.2	Requirements	7
1.3	Procedure	7
1.4	Conclusion	7
1.5	Precautions	7
2	Deletion of a database	8
2.1	Aim	8
2.2	Requirements	8
2.3	Procedure	8
2.4	Conclusion	8
2.5	Precautions	8
3	Creation of a table	9
3.1	Aim	9
3.2	Requirements	9
3.3	Procedure	9
3.4	Conclusion	9
3.5	Precautions	9
4	Show the databases and tables	10
4.1	Aim	10
4.2	Requirements	10
4.3	Procedure	10
4.4	Conclusion	10
4.5	Precautions	10
5	List out the field name along with its field type and constraints for a table.	11
5.1	Aim	11
5.2	Requirements	11
5.3	Procedure	11
5.4	Conclusion	11
5.5	Precautions	11
6	Alter a table	12
6.1	Aim	12
6.2	Requirements	12
6.3	Procedure	12
6.4	Conclusion	12
6.5	Precautions	12
7	Create primary key constraints	13
7.1	Aim	13
7.2	Requirements	13
7.3	Procedure	13
7.4	Conclusion	13
7.5	Precautions	13
8	Set a default value for a field	14
8.1	Aim	14
8.2	Requirements	14
8.3	Procedure	14
8.4	Conclusion	14
8.5	Precautions	14

9 Delete a Table	15
9.1 Aim	15
9.2 Requirements	15
9.3 Procedure	15
9.4 Conclusion	15
9.5 Precautions	15
10 Foreign Key Constraints	16
10.1 Aim	16
10.2 Requirements	16
10.3 Procedure	16
10.4 Conclusion	16
10.5 Precautions	17
11 Insert records into a table	18
11.1 Aim	18
11.2 Requirements	18
11.3 Procedure	18
11.4 Conclusion	19
11.5 Precautions	19
12 Display data from the database Part I	20
12.1 Aim	20
12.2 Requirements	20
12.3 Procedure	20
12.4 Conclusion	21
12.5 Precautions	21
13 Display data from the database Part II	22
13.1 Aim	22
13.2 Requirements	22
13.3 Procedure	22
13.4 Conclusion	23
13.5 Precautions	23
14 Advanced Queries to Display Data from the Database	24
14.1 Aim	24
14.2 Requirements	24
14.3 Procedure	24
14.4 Conclusion	25
14.5 Precautions	25
15 Nested Queries in MySQL	26
15.1 Aim	26
15.2 Requirements	26
15.3 Procedure	26
15.4 Conclusion	26
15.5 Precautions	27
16 Create a Java Program to Demonstrate Variables and Data Types	29
16.1 Aim	29
16.2 Requirements	29
16.3 Procedure	29
16.4 Conclusion	29
16.5 Precautions	30

17 Create a Java Program to Demonstrate Control Flow	31
17.1 Aim	31
17.2 Requirements	31
17.3 Procedure	31
17.4 Conclusion	31
17.5 Precautions	32
18 Create a Java Program to Demonstrate switch-case Statement	33
18.1 Aim	33
18.2 Requirements	33
18.3 Procedure	33
18.4 Conclusion	33
18.5 Precautions	34
19 Create a Java Program to Demonstrate while Loop	35
19.1 Aim	35
19.2 Requirements	35
19.3 Procedure	35
19.4 Conclusion	35
19.5 Precautions	36
20 Create a Java Program to Demonstrate for for Loop and Arrays	37
20.1 Aim	37
20.2 Requirements	37
20.3 Procedure	37
20.4 Conclusion	37
20.5 Precautions	38
21 Create a Java Program to Demonstrate Methods	39
21.1 Aim	39
21.2 Requirements	39
21.3 Procedure	39
21.4 Conclusion	39
21.5 Precautions	39
22 Create a Java Program to Demonstrate OOP	40
22.1 Aim	40
22.2 Requirements	40
22.3 Procedure	40
22.4 Conclusion	41
22.5 Precautions	41
23 Create a Java Program to Demonstrate Access Modifiers and GettersSetters	42
23.1 Aim	42
23.2 Requirements	42
23.3 Procedure	42
23.4 Conclusion	43
23.5 Precautions	43
24 Create a Java Program to Demonstrate Data Input using Scanner Class	44
24.1 Aim	44
24.2 Requirements	44
24.3 Procedure	44
24.4 Conclusion	45
24.5 Precautions	45

25 Create a Java Program to Demonstrate Arrays and String Manipulation	46
25.1 Aim	46
25.2 Requirements	46
25.3 Procedure	46
25.4 Conclusion	47
25.5 Precautions	47
26 Create a Java Program to Demonstrate Exception Handling	48
26.1 Aim	48
26.2 Requirements	48
26.3 Procedure	48
26.4 Conclusion	49
26.5 Precautions	49
27 Create a Java Program to Demonstrate File Database Connectivity using JDBC	50
27.1 Aim	50
27.2 Requirements	50
27.3 Procedure	50
27.4 Conclusion	51
27.5 Precautions	51
28 Create a Java Program to Demonstrate Assertions	52
28.1 Aim	52
28.2 Requirements	52
28.3 Procedure	52
28.4 Conclusion	53
28.5 Precautions	53
29 Create a Java Program to Demonstrate Multithreading	54
29.1 Aim	54
29.2 Requirements	54
29.3 Procedure	54
29.4 Conclusion	55
29.5 Precautions	55
30 Create a Java Program to Demonstrate Wrapper Classes	56
30.1 Aim	56
30.2 Requirements	56
30.3 Procedure	56
30.4 Conclusion	57
30.5 Precautions	57

Unit - 1: Database Concepts

1 Creation of a Database

1.1 Aim

To create the School database in MySQL.

1.2 Requirements

- MySQL Community Server installed and running.
- MySQL Command Line Client or any MySQL GUI tool (e.g., MySQL Workbench).
- MySQL user credentials (username and password).
- Basic knowledge of SQL commands.

1.3 Procedure

1. Open the MySQL Command Line Client.
2. Log in to the MySQL server using your credentials.

(a) Enter terminal and type the command:

```
mysql -h mysql.oavbibina.local -u username -p # Replace 'username' with your MySQL username
```

Listing 1: Open MySQL Command Line

- (b) Press Enter and type your root password when prompted.
(c) You should see the MySQL prompt (e.g., mysql>).

3. Create the new School database using the SQL command:

```
CREATE DATABASE School;
```

Listing 2: Creating School Database in MySQL.

1.4 Conclusion

The **School** database was successfully created, providing a container for subsequent table creation and data management.

1.5 Precautions

- Ensure you have the necessary privileges to create a database.
- Use a unique name for the database to avoid conflicts with existing databases.

2 Deletion of a database

2.1 Aim

To delete the School database in MySQL.

2.2 Requirements

- MySQL Community Server installed and running.
- MySQL Command Line Client or any MySQL GUI tool (e.g., MySQL Workbench).
- MySQL user credentials (username and password).
- Basic knowledge of SQL commands.

2.3 Procedure

1. Open the MySQL Command Line Client.
2. Log in to the MySQL server using your credentials.

(a) Enter terminal and type the command:

```
mysql -h mysql.oavbibina.local -u username -p # Replace 'username' with your MySQL
username
```

Listing 3: Open MySQL Command Line

- (b) Press Enter and type your root password when prompted.
(c) You should see the MySQL prompt (e.g., mysql>).

3. Delete the existing School database using the SQL command:

```
DROP DATABASE School;
```

Listing 4: Deleting School Database in MySQL.

2.4 Conclusion

The **School** database was successfully removed from the MySQL server, demonstrating the database deletion process.

2.5 Precautions

- Ensure you have the necessary privileges to delete a database.
- Be cautious when deleting databases, as this action is irreversible and all data will be lost.
- Make sure the **School** database exists before attempting to delete it.

3 Creation of a table

3.1 Aim

To create **Teacher** table in the **School** database.

3.2 Requirements

- MySQL Community Server installed and running.
- MySQL Command Line Client or any MySQL GUI tool (e.g., MySQL Workbench).
- MySQL user credentials (username and password).
- Basic knowledge of SQL commands.

3.3 Procedure

1. Open the MySQL Command Line Client.
2. Log in to the MySQL server using your credentials.

(a) Enter terminal and type the command:

```
mysql -h mysql.oavbibina.local -u username -p # Replace 'username' with your MySQL username
```

Listing 5: Open MySQL Command Line

(b) Press Enter and type your root password when prompted.

(c) You should see the MySQL prompt (e.g., mysql>).

3. Create the TEACHER table using the SQL command:

```
CREATE DATABASE School; -- Recreate the School database
USE School; -- Select the School database
CREATE TABLE Teacher (
    Teacher_ID INTEGER,
    First_Name VARCHAR(20),
    Last_Name VARCHAR(20),
    Gender CHAR(1),
    Salary DECIMAL(10,2),
    Date_of_Birth DATE,
    Dept_No INTEGER
);
```

Listing 6: Creating TEACHER Table in MySQL.

3.4 Conclusion

The **Teacher** table was successfully created within the **School** database with the specified columns, establishing the structure for storing teacher information.

3.5 Precautions

- Ensure you are using the correct database before creating the table.
- Use appropriate data types for each column based on the expected data.
- Avoid using reserved keywords as column names (e.g., ID, Name).

4 Show the databases and tables

4.1 Aim

To show the databases and tables in MySQL.

4.2 Requirements

- MySQL Community Server installed and running.
- MySQL Command Line Client or any MySQL GUI tool (e.g., MySQL Workbench).
- MySQL user credentials (username and password).
- Basic knowledge of SQL commands.

4.3 Procedure

1. Open the MySQL Command Line Client.
2. Log in to the MySQL server using your credentials.

(a) Enter terminal and type the command:

```
mysql -h mysql.oavbibina.local -u username -p # Replace 'username' with your MySQL username
```

Listing 7: Open MySQL Command Line

(b) Press Enter and type your root password when prompted.

(c) You should see the MySQL prompt (e.g., mysql>).

3. Show all databases using the SQL command:

```
SHOW DATABASES;
```

Listing 8: Show Databases in MySQL.

4. Show all tables in the current database using the SQL command:

```
USE School; -- Select the School database  
SHOW TABLES;
```

Listing 9: Show Tables in MySQL.

4.4 Conclusion

The available databases and the tables within the **School** database (specifically the **Teacher** table) were successfully listed, confirming their existence.

4.5 Precautions

- Ensure you are using the correct database before showing the tables.
- Use appropriate commands to display the databases and tables.
- Be cautious with user credentials and permissions.

5 List out the field name along with its field type and constraints for a table.

5.1 Aim

To list out the field name along with its field type and constraints for the **Teacher** table in MySQL.

5.2 Requirements

- MySQL Community Server installed and running.
- MySQL Command Line Client or any MySQL GUI tool (e.g., MySQL Workbench).
- MySQL user credentials (username and password).
- Basic knowledge of SQL commands.

5.3 Procedure

1. Open the MySQL Command Line Client.
2. Log in to the MySQL server using your credentials.

(a) Enter terminal and type the command:

```
mysql -h mysql.oavbibina.local -u username -p # Replace 'username' with your MySQL username
```

Listing 10: Open MySQL Command Line

- (b) Press Enter and type your root password when prompted.
 - (c) You should see the MySQL prompt (e.g., mysql>).
3. Show the field name along with its field type and constraints for the **Teacher** table using the SQL command:

```
USE School; -- Select the School database
DESCRIBE Teacher; -- Show field names and types for the Teacher table
DESC Teacher; -- Show field names and types for the Teacher table
```

Listing 11: Show Field Names and Types in MySQL.

5.4 Conclusion

The structure of the **Teacher** table, including field names, data types, and constraints, was successfully displayed using the 'DESCRIBE' command.

5.5 Precautions

- Ensure you are using the correct database before showing the field names and types.

6 Alter a table

6.1 Aim

To alter the structure of the **Teacher** table to make **First_Name** field as NOT NULL.

6.2 Requirements

- MySQL Community Server installed and running.
- MySQL Command Line Client or any MySQL GUI tool (e.g., MySQL Workbench).
- MySQL user credentials (username and password).
- Basic knowledge of SQL commands.

6.3 Procedure

1. Open the MySQL Command Line Client.
2. Log in to the MySQL server using your credentials and select **Teacher** table.

(a) Enter terminal and type the command:

```
mysql -h mysql.oavbibina.local -u username -p # Replace 'username' with your MySQL
username
USE School; -- Select the School database
```

Listing 12: Open MySQL Command Line

- (b) Press Enter and type your root password when prompted.
(c) You should see the MySQL prompt (e.g., mysql>).

3. Alter the **Teacher** table to make **First_Name** field as NOT NULL using the SQL command:

```
ALTER TABLE Teacher
MODIFY First_Name VARCHAR(20) NOT NULL;
```

Listing 13: Altering TEACHER Table in MySQL.

4. Show the field name along with its field type and constraints for the **Teacher** table using the SQL command:

```
USE School; -- Select the School database
DESCRIBE Teacher; -- Show field names and types for the Teacher table
DESC Teacher; -- Show field names and types for the Teacher table
```

Listing 14: Show Field Names and Types in MySQL.

6.4 Conclusion

The structure of the **Teacher** table was successfully modified by adding a NOT NULL constraint to the **First_Name** column, enforcing data integrity.

6.5 Precautions

- Ensure you are using the correct database before altering the table.
- Use appropriate commands to alter the table structure.
- Be cautious with user credentials and permissions.
- Ensure that the data in the column being modified is compatible with the new constraints.

7 Create primary key constraints

7.1 Aim

To alter the structure of the **Teacher** table to make **Teacher_ID** field as PRIMARY KEY.

7.2 Requirements

- MySQL Community Server installed and running.
- MySQL Command Line Client or any MySQL GUI tool (e.g., MySQL Workbench).
- MySQL user credentials (username and password).
- Basic knowledge of SQL commands.

7.3 Procedure

1. Open the MySQL Command Line Client.
2. Log in to the MySQL server using your credentials and select **Teacher** table.

(a) Enter terminal and type the command:

```
mysql -h mysql.oavbibina.local -u username -p # Replace 'username' with your MySQL
username
USE School; -- Select the School database
```

Listing 15: Open MySQL Command Line

(b) Press Enter and type your root password when prompted.

(c) You should see the MySQL prompt (e.g., mysql>).

3. Alter the **Teacher** table to make the **Teacher_ID** field the PRIMARY KEY using the SQL command:

```
ALTER TABLE Teacher
MODIFY Teacher_ID INTEGER PRIMARY KEY;
```

Listing 16: Altering TEACHER Table in MySQL.

4. Show the field name along with its field type and constraints for the **Teacher** table using the SQL command:

```
USE School; -- Select the School database
DESCRIBE Teacher; -- Show field names and types for the Teacher table
DESC Teacher; -- Show field names and types for the Teacher table
```

Listing 17: Show Field Names and Types in MySQL.

7.4 Conclusion

The structure of the **Teacher** table was successfully modified by adding a PRIMARY KEY constraint to the **Teacher_ID** column, ensuring unique identification for each teacher record.

7.5 Precautions

- Ensure you are using the correct database before altering the table.

8 Set a default value for a field

8.1 Aim

To alter the structure of the **Teacher** table to set a default value of Rs. 40,000 for the **Salary** field.

8.2 Requirements

- MySQL Community Server installed and running.
- MySQL Command Line Client or any MySQL GUI tool (e.g., MySQL Workbench).
- MySQL user credentials (username and password).
- Basic knowledge of SQL commands.

8.3 Procedure

1. Open the MySQL Command Line Client.
2. Log in to the MySQL server using your credentials and select **Teacher** table.

(a) Enter terminal and type the command:

```
mysql -h mysql.oavbibina.local -u username -p # Replace 'username' with your MySQL
username
USE School; -- Select the School database
```

Listing 18: Open MySQL Command Line

- (b) Press Enter and type your root password when prompted.
- (c) You should see the MySQL prompt (e.g., mysql>).

3. Alter the **Teacher** table to set a default value for the **Salary** field using the SQL command:

```
ALTER TABLE Teacher
MODIFY Salary DECIMAL(10,2) DEFAULT 40000; -- Set default value for Salary field
```

Listing 19: Altering TEACHER Table in MySQL.

4. Show the field name along with its field type and constraints for the **Teacher** table using the SQL command:

```
USE School; -- Select the School database
DESCRIBE Teacher; -- Show field names and types for the Teacher table
DESC Teacher; -- Show field names and types for the Teacher table
```

Listing 20: Show Field Names and Types in MySQL.

8.4 Conclusion

The structure of the **Teacher** table was successfully modified by adding a default value of 40000 for the **Salary** column, simplifying data entry for new records.

8.5 Precautions

- Ensure you are using the correct database before altering the table.
- Use appropriate commands to alter the table structure.
- Be cautious with user credentials and permissions.
- Ensure that the data in the column being modified is compatible with the new constraints.

9 Delete a Table

9.1 Aim

To delete the **Teacher** table from the **School** database.

9.2 Requirements

- MySQL Community Server installed and running.
- MySQL Command Line Client or any MySQL GUI tool (e.g., MySQL Workbench).
- MySQL user credentials (username and password).
- Basic knowledge of SQL commands.

9.3 Procedure

1. Open the MySQL Command Line Client.
2. Log in to the MySQL server using your credentials and select **School** database.
 - (a) Enter terminal and type the command:

```
mysql -h mysql.oavbibina.local -u username -p # Replace 'username' with your MySQL
username
USE School; -- Select the School database
```

Listing 21: Open MySQL Command Line

- (b) Press Enter and type your root password when prompted.
 - (c) You should see the MySQL prompt (e.g., mysql>).
3. Delete the **Teacher** table using the SQL command:

```
DROP TABLE Teacher; -- Delete the Teacher table
```

Listing 22: Deleting TEACHER Table in MySQL.

9.4 Conclusion

The structure of the **Teacher** table was successfully modified by deleting it from the **School** database, demonstrating the table deletion process.

9.5 Precautions

- Ensure that the table (Teacher) exists before attempting to delete it.
- Be cautious when deleting tables, as this action is irreversible and all data will be lost.
- Be cautious with user credentials and permissions.

10 Foreign Key Constraints

10.1 Aim

To create a foreign key constraint in the **Teacher** table that references the **Department** table.

10.2 Requirements

- MySQL Community Server installed and running.
- MySQL Command Line Client or any MySQL GUI tool (e.g., MySQL Workbench).
- MySQL user credentials (username and password).
- Basic knowledge of SQL commands.

10.3 Procedure

1. Open the MySQL Command Line Client.
2. Log in to the MySQL server using your credentials.

- (a) Enter terminal and type the command:

```
mysql -h mysql.oavbibina.local -u username -p # Replace 'username' with your MySQL
username
```

Listing 23: Open MySQL Command Line

- (b) Press Enter and type your root password when prompted.
- (c) You should see the MySQL prompt (e.g., mysql>).

3. Create the **Department** table with a primary key.

```
USE School; -- Select the School database
CREATE TABLE Department (
    Dept_ID INTEGER PRIMARY KEY,
    Dept_Name VARCHAR(20) NOT NULL
);
```

Listing 24: Creating DEPARTMENT Table in MySQL.

4. Create the **Teacher** table with a foreign key constraint referencing the **Department** table.

```
CREATE TABLE Teacher (
    Teacher_ID INTEGER PRIMARY KEY,
    First_Name VARCHAR(20) NOT NULL,
    Last_Name VARCHAR(20),
    Gender CHAR(1),
    Salary DECIMAL(10,2) DEFAULT 40000,
    Date_of_Birth DATE,
    Dept_No INTEGER,
    FOREIGN KEY (Dept_No) REFERENCES Department(Dept_ID) ON DELETE SET NULL ON UPDATE
    NULL
);
```

Listing 25: Creating TEACHER Table with Foreign Key in MySQL.

10.4 Conclusion

A foreign key constraint was successfully established between the **Teacher** and **Department** tables, enforcing referential integrity by linking **Dept_No** in **Teacher** to **Dept_ID** in **Department**.

10.5 Precautions

- Ensure that the referenced table (Department) exists before creating the foreign key constraint.
- Use appropriate data types for the foreign key and primary key columns to ensure compatibility.
- Be cautious with user credentials and permissions.
- Ensure you are using the correct database before showing the field names and types.

11 Insert records into a table

11.1 Aim

Add the following records into the **Department** and **Teacher** table in MySQL.

Table 1: Department Table Records.

Dept_ID	Dept_Name
1	Physics
2	Computer Science
3	English
4	Hindi

Table 2: Teacher Table Records.

Teacher_ID	First_Name	Last_Name	Gender	Salary	Date_of_Birth	Dept_No
101	Shanaya	Batra	F	50000	1984-08-11	1
102	Alice	Walton	F	48000	1983-02-12	2
103	Surbhi	Bansal	F	34000	1985-06-11	4
106	Tarun	Mehta	M	50000	1980-08-21	2
107	Puneet	NULL	M	52500	1976-09-25	3

11.2 Requirements

- MySQL Community Server installed and running.
- MySQL Command Line Client or any MySQL GUI tool (e.g., MySQL Workbench).
- MySQL user credentials (username and password).
- Basic knowledge of SQL commands.

11.3 Procedure

1. Open the MySQL Command Line Client.
2. Log in to the MySQL server using your credentials.

(a) Enter terminal and type the command:

```
mysql -h mysql.oavbibina.local -u username -p # Replace 'username' with your MySQL
username
```

Listing 26: Open MySQL Command Line

(b) Press Enter and type your root password when prompted.

(c) You should see the MySQL prompt (e.g., mysql>).

3. Insert the following records into the **Department** table:

```
USE School; -- Select the School database
INSERT INTO Department (Dept_ID, Dept_Name) VALUES
(1, 'Physics'),
(2, 'Computer Science'),
(3, 'English'),
(4, 'Hindi');
```

Listing 27: Inserting Records into DEPARTMENT Table in MySQL.

4. Insert the records in the Table 2 into the **Teacher** table:

```
INSERT INTO Teacher VALUES
  (101, 'Shanaya', 'Batra', 'F', '50000', '1984-08-11', '1'),
  (102, 'Alice', 'Walton', 'F', '48000', '1983-02-12', '2'),
  (103, 'Surbhi', 'Bansal', 'F', '34000', '1985-06-11', '4'),
  (106, 'Tarun', 'Mehta', 'M', 50000, '1980-08-21', 2),
  (107, 'Puneet', NULL, 'M', 52500, '1976-09-25', 3);
```

Listing 28: Inserting Records into TEACHER Table in MySQL.

11.4 Conclusion

Data records were successfully inserted into both the **Department** and **Teacher** tables, populating them with initial information.

11.5 Precautions

- Ensure that the MySQL server is running before executing the commands.
- Double-check the syntax of your SQL commands to avoid errors.
- Ensure that the data types of the values being inserted match the column definitions in the table.
- Be cautious with user credentials and permissions.
- Ensure that the foreign key constraints are satisfied when inserting records into the **Teacher** table.
- Ensure that the records being inserted do not violate any constraints defined on the table (e.g., NOT NULL, UNIQUE).

12 Display data from the database Part I

12.1 Aim

To display the following data from the **Teacher** table and **Department** table from the **School** database in MySQL.

Table 3: Teacher Table Records.

Teacher_ID	First_Name	Last_Name	Gender	Salary	Date_of_Birth	Dept_No
101	Shanaya	Batra	F	50000	1984-08-11	1
102	Alice	Walton	F	48000	1983-02-12	2
103	Surbhi	Bansal	F	34000	1985-06-11	4
106	Tarun	Mehta	M	50000	1980-08-21	2
107	Puneet	NULL	M	52500	1976-09-25	3

- To retrieve all the information about Teacher with ID=101.
- To find the names of all teachers earning more than 50000.
- To display Teacher_ID, First_Name, Last_Name and Dept_No of teachers who belong to department number 2 or 3.
- To retrieve names of all the teachers and the names and id of their respective departments.
- To retrieve names of all the teachers and the names and id of their respective departments where department name is either 'Computer Science' or 'Hindi'.

12.2 Requirements

- MySQL Community Server installed and running.
- MySQL Command Line Client or any MySQL GUI tool (e.g., MySQL Workbench).
- MySQL user credentials (username and password).
- Basic knowledge of SQL commands.

12.3 Procedure

1. Open the MySQL Command Line Client.
2. Log in to the MySQL server using your credentials and select the **School** database.

(a) Enter terminal and type the command:

```
mysql -h mysql.oavbibina.local -u username -p # Replace 'username' with your MySQL
username
USE School; -- Select the School database
```

Listing 29: Open MySQL Command Line

(b) Press Enter and type your root password when prompted.

(c) You should see the MySQL prompt (e.g., mysql>).

3. Display all records from the **Teacher** table using the SQL command:

```
SELECT * FROM Teacher; -- Display all records from the Teacher table
```

Listing 30: Display Data from TEACHER Table in MySQL.

4. Display all records from the **Department** table using the SQL command:

```
SELECT * FROM Department; -- Display all records from the Department table
```

Listing 31: Display Data from DEPARTMENT Table in MySQL.

5. Run different **Query** commands to display the data from the database:

- (a) To retrieve all the information about Teacher with ID=101:

```
SELECT * FROM Teacher
WHERE Teacher_ID = 101; -- Display all information about Teacher with ID=101
```

Listing 32: Display Data from TEACHER Table in MySQL.

- (b) To find the names of all teachers earning more than 50000:

```
SELECT First_Name, Last_Name FROM Teacher
WHERE Salary > 50000; -- Find names of teachers earning more than 50000
```

Listing 33: Display Data from TEACHER Table in MySQL.

- (c) To display Teacher_ID, First_Name, Last_Name and Dept_No of teachers who belong to department number 4 or 7:

```
SELECT Teacher_ID, First_Name, Last_Name, Dept_No FROM Teacher
WHERE Dept_No IN (2, 3); -- Display Teacher_ID, First_Name, Last_Name and
Dept_No of teachers who belong to department number 2 or 3
```

Listing 34: Display Data from TEACHER Table in MySQL.

- (d) To retrieve names of all the teachers and the names and id of their respective departments.

```
SELECT First_Name, Last_Name, Dept_ID, Dept_Name
FROM Teacher, Department
WHERE Department.Dept_ID=Teacher.Dept_No; -- Retrieve names of all the teachers
and the names and id of their respective departments
```

Listing 35: Display Data from TEACHER Table in MySQL.

- (e) To retrieve names of all the teachers and the names and id of their respective departments where department name is either 'Computer Science' or 'Hindi':

```
SELECT First_Name, Last_Name, Dept_ID, Dept_Name
FROM Teacher, Department
WHERE Department.Dept_ID=Teacher.Dept_No
AND (Department.Dept_Name='Computer Science'
OR Department.Dept_Name='Hindi'); -- Retrieve names of all the teachers and
the names and id of their respective departments where department name is either
'Computer Science' or 'Hindi'
```

Listing 36: Display Data from TEACHER Table in MySQL.

12.4 Conclusion

Various **SELECT** queries were successfully executed to retrieve specific subsets of data from the **Teacher** and **Department** tables based on different criteria, demonstrating basic data retrieval techniques.

12.5 Precautions

- Ensure that the MySQL server is running before executing the commands.

13 Display data from the database Part II

13.1 Aim

To display the following data from the **Teacher** table and **Department** table from the **School** database in MySQL.

Table 4: Teacher Table Records.

Teacher_ID	First_Name	Last_Name	Gender	Salary	Date_of_Birth	Dept_No
101	Shanaya	Batra	F	50000	1984-08-11	1
102	Alice	Walton	F	48000	1983-02-12	2
103	Surbhi	Bansal	F	34000	1985-06-11	4
106	Tarun	Mehta	M	50000	1980-08-21	2
107	Puneet	NULL	M	52500	1976-09-25	3

- To list distinct values of an attribute in a table.
- To match part of the string.
- To sort the result of a query based on some attributes.
- To test whether a value is unavailable or unknown or not applicable in a column (i.e, NULL values)

13.2 Requirements

- MySQL Community Server installed and running.
- MySQL Command Line Client or any MySQL GUI tool (e.g., MySQL Workbench).
- MySQL user credentials (username and password).
- Basic knowledge of SQL commands.

13.3 Procedure

1. Open the MySQL Command Line Client.
2. Log in to the MySQL server using your credentials and select the **School** database.

- (a) Enter terminal and type the command:

```
mysql -h mysql.oavbibina.local -u username -p # Replace 'username' with your MySQL
username
USE School; -- Select the School database
```

Listing 37: Open MySQL Command Line

- (b) Press Enter and type your root password when prompted.
- (c) You should see the MySQL prompt (e.g., mysql>).

3. Run different **Query** commands to display the data from the database:

- (a) To list distinct values of an attribute in a table:

```
SELECT DISTINCT Dept_No
FROM Teacher
WHERE Gender = 'M'; -- List distinct values of Dept_No in Teacher table where
```

Listing 38: Display Data from TEACHER Table in MySQL.

- (b) To match part of the string.

```
SELECT First_Name, Last_Name FROM Teacher
WHERE First_Name LIKE "S%"; -- retrieve names of all the teachers starting from
letter 'S'.
```

Listing 39: Display Data from TEACHER Table in MySQL.

```
SELECT First_Name
FROM Teacher
WHERE First_Name LIKE "S_____"; -- To retrieve names of all the teachers having 6
characters in the first name and starting with 'S'.
```

Listing 40: Display Data from TEACHER Table in MySQL.

```
SELECT First_Name
FROM Teacher
WHERE First_Name LIKE "_____%"; -- retrieve names of all the teachers having
at least 6 characters in the first name.
```

Listing 41: Display Data from TEACHER Table in MySQL.

- (c) To sort the result of a query based on some attributes.

```
SELECT First_Name, Last_Name
FROM Teacher
ORDER BY First_Name, Last_Name; -- To retrieve names of all the teachers in
ascending order of their first name and last name.
```

Listing 42: Display Data from TEACHER Table in MySQL.

```
SELECT First_Name, Last_Name, Dept_No
FROM Teacher
ORDER BY Dept_No ASC, First_Name DESC, Last_Name DESC; -- To retrieve names of
all the teachers in ascending order of their department number and in descending
order of their first name and last name.
```

Listing 43: Display Data from TEACHER Table in MySQL.

- (d) To test whether a value is unavailable or unknown or not applicable in a column (i.e, NULL values)

```
SELECT *
FROM Teacher
WHERE Last_Name IS NULL; -- To retrieve names of all the teachers having NULL
values in the last name.
```

Listing 44: Display Data from TEACHER Table in MySQL.

13.4 Conclusion

Data retrieval was further explored using ‘**DISTINCT**’, ‘**LIKE**’, ‘**ORDER BY**’, and ‘**IS NULL**’ clauses, showcasing techniques for filtering, pattern matching, sorting, and handling NULL values in SQL queries.

13.5 Precautions

- Ensure that the MySQL server is running before executing the commands.

14 Advanced Queries to Display Data from the Database

14.1 Aim

The aim is to display the following data from the **Teacher** table (and potentially the **Department** table, although not explicitly queried below) within the **School** database in MySQL using advanced queries.

Table 5: Teacher Table Records.

Teacher_ID	First_Name	Last_Name	Gender	Salary	Date_of_Birth	Dept_No
101	Shanaya	Batra	F	50000	1984-08-11	1
102	Alice	Walton	F	48000	1983-02-12	2
103	Surbhi	Bansal	F	34000	1985-06-11	4
106	Tarun	Mehta	M	50000	1980-08-21	2
107	Puneet	NULL	M	52500	1976-09-25	3

- Count the number of teachers in each department.
- Find the sum of salaries of all teachers in each department.
- Find the maximum salary among teachers in each department.
- Find the minimum salary among teachers in each department.
- Find the average salary of teachers in each department.
- Calculate the new salaries of teachers by increasing their current salary by 10%.

14.2 Requirements

- MySQL Community Server installed and running.
- MySQL Command Line Client or any MySQL GUI tool (e.g., MySQL Workbench).
- MySQL user credentials (username and password).
- Basic knowledge of SQL commands.

14.3 Procedure

1. Open the MySQL Command Line Client.
2. Log in to the MySQL server using your credentials and select the **School** database.

(a) Open a terminal and type the following command:

```
mysql -h mysql.oavbibina.local -u username -p # Replace 'username' with your MySQL
username
USE School; -- Select the School database
```

Listing 45: Open MySQL Command Line

- (b) Press Enter and type your password when prompted.
 - (c) You should see the MySQL prompt (e.g., mysql>).
3. Execute the following SQL queries to apply aggregate functions and perform calculations:
 - (a) To count the number of teachers in each department:

```
SELECT Dept_No, COUNT(*) AS Number_of_Teachers
FROM Teacher
GROUP BY Dept_No; -- Counts the number of teachers in each department
```

Listing 46: Count Teachers per Department

- (b) To find the sum of salaries of all teachers in each department:

```
SELECT Dept_No , SUM(Salary) AS Total_Salary
FROM Teacher
GROUP BY Dept_No; -- Sum of salaries of all teachers in each department
```

Listing 47: Sum of Salaries per Department

- (c) To find the maximum salary of teachers in each department:

```
SELECT Dept_No , MAX(Salary) AS Max_Salary
FROM Teacher
GROUP BY Dept_No; -- Maximum salary of teachers in each department
```

Listing 48: Maximum Salary per Department

- (d) To find the minimum salary of teachers in each department:

```
SELECT Dept_No , MIN(Salary) AS Min_Salary
FROM Teacher
GROUP BY Dept_No; -- Minimum salary of teachers in each department
```

Listing 49: Minimum Salary per Department

- (e) To find the average salary of teachers in each department:

```
SELECT Dept_No , AVG(Salary) AS Avg_Salary
FROM Teacher
GROUP BY Dept_No; -- Average salary of teachers in each department
```

Listing 50: Average Salary per Department

- (f) To calculate new salaries of teachers by increasing the current salary by 10%:

```
SELECT Teacher_ID , First_Name , Last_Name , Salary , Salary * 1.10 AS New_Salary
FROM Teacher; -- Calculate new salaries of teachers by increasing the current
salary by 10%
```

Listing 51: Calculate New Salary (10% Increase)

14.4 Conclusion

These advanced queries successfully demonstrate the use of SQL aggregate functions (**COUNT**, **SUM**, **MAX**, **MIN**, **AVG**) to perform calculations on groups of records based on the department. The results effectively show the number of teachers, total salary, maximum salary, minimum salary, and average salary per department, as well as the projected new salaries after a 10% increase for each teacher in the **Teacher** table.

14.5 Precautions

- Ensure the MySQL server is running before executing commands.
- Double-check the syntax of SQL commands to avoid errors.
- Handle user credentials and database permissions securely.
- Verify that the data types used in calculations are compatible with the column definitions.
- Be mindful of potential calculation issues like overflow or division by zero (if applicable in other queries).

15 Nested Queries in MySQL

15.1 Aim

To demonstrate the use of nested queries (subqueries) to retrieve data from the **Teacher** and **Department** tables within the **School** database in MySQL.

15.2 Requirements

- MySQL Community Server installed and running.
- MySQL Command Line Client or any MySQL GUI tool (e.g., MySQL Workbench).
- MySQL user credentials (username and password).
- Basic knowledge of SQL commands.

15.3 Procedure

1. Open the MySQL Command Line Client.
2. Log in to the MySQL server using your credentials and select the database.

- (a) Enter terminal and type the command:

```
mysql -h mysql.oavbibina.local -u username -p # Replace 'username' with your MySQL
username
USE School; -- Select the School database
```

Listing 52: Open MySQL Command Line

- (b) Press Enter and type your password when prompted.
- (c) You should see the MySQL prompt (e.g., mysql>).

3. Execute the following SQL nested queries to display data from the database:

- (a) Find the names of teachers whose salary is greater than the overall average salary:

```
SELECT First_Name, Last_Name, Salary
FROM Teacher
WHERE Salary > (SELECT AVG(Salary) FROM Teacher);
```

Listing 53: Subquery: Greater than Average Salary

- (b) Find the details of teachers working in the 'Computer Science' department utilizing an **IN** clause subquery:

```
SELECT First_Name, Last_Name, Dept_No
FROM Teacher
WHERE Dept_No IN (SELECT Dept_ID FROM Department WHERE Dept_Name = 'Computer
Science');
```

Listing 54: Subquery: Filter by Department

- (c) Find the teacher(s) having the highest salary:

```
SELECT First_Name, Last_Name, Salary
FROM Teacher
WHERE Salary = (SELECT MAX(Salary) FROM Teacher);
```

Listing 55: Subquery: Maximum Salary

15.4 Conclusion

This experiment demonstrates how to use nested queries (subqueries) in MySQL to perform complex data retrieval by dynamically evaluating inner conditions before applying them to outer queries.

15.5 Precautions

- Ensure the MySQL server is running before executing commands.
- Ensure the inner query meant for `=` comparison returns only a single value, not multiple rows.
- Use **IN** or **ANY** when the inner query might return multiple rows.
- Always encapsulate nested subqueries inside proper parentheses.

Unit - 3: Fundamentals of Java Programming

16 Create a Java Program to Demonstrate Variables and Data Types

16.1 Aim

To create a Java program that calculates percentage marks obtained by a student.

16.2 Requirements

- Apache NetBeans IDE installed on your computer.
- Java Development Kit (JDK) installed on your computer.
- Basic knowledge of Java programming.

16.3 Procedure

1. Open Apache NetBeans IDE on your computer.
2. Create a new Java project:
 - Click on **File** → **New Project**.
 - In the New Project dialog, select **Java** from the Categories list and **Java Application** from the Projects list. Click **Next**.
 - Enter a project name (e.g., "PercentageCalculator") and choose a location to save the project. Ensure that the "Create Main Class" option is checked. Click **Finish**.
3. Write the Java program:
 - In the Project Explorer, expand the project folder and locate the main class file (e.g., PercentageCalculator.java).
 - Double-click on the main class file to open it in the editor.
 - Replace the existing code with the following code:

```
public class PercentageCalculator {  
    public static void main(String[] args) {  
        String name = "Alice";  
        int totalMarks = 500;  
        int obtainedMarks = 425;  
  
        double percentage = ((double) obtainedMarks / totalMarks) * 100;  
  
        System.out.println(name + " obtained " + percentage + "% marks.");  
    }  
}
```

Listing 56: Percentage Calculator Java Program

4. Run the Java program:
 - Click on the **Run** menu and select **Run Project** (or press F6).
 - The output window will display the percentage marks obtained by the student.

16.4 Conclusion

The Java program successfully demonstrates the use of variables and data types to calculate and display the percentage marks obtained by a student. This exercise highlights the importance of data types in performing arithmetic operations and storing values in Java.

16.5 Precautions

- Ensure that Apache NetBeans IDE and JDK are properly installed and configured on your computer.
- Double-check the code for syntax errors before running the program.
- Save your work frequently to avoid losing any changes.
- Use meaningful names for variables to enhance code readability.

17 Create a Java Program to Demonstrate Control Flow

17.1 Aim

To create a Java programme to check if a configurable year is a leap year using if-else statement.

17.2 Requirements

- Apache NetBeans IDE installed on your computer.
- Java Development Kit (JDK) installed on your computer.
- Basic knowledge of Java programming.

17.3 Procedure

1. Open Apache NetBeans IDE on your computer.
2. Create a new Java project:
 - Click on **File** → **New Project**.
 - In the New Project dialog, select **Java** from the Categories list and **Java Application** from the Projects list. Click **Next**.
 - Enter a project name (e.g., "LeapYearChecker") and choose a location to save the project. Ensure that the "Create Main Class" option is checked. Click **Finish**.
3. Write the Java program:
 - In the Project Explorer, expand the project folder and locate the main class file (e.g., LeapYearChecker.java).
 - Double-click on the main class file to open it in the editor.
 - Replace the existing code with the following code:

```
public class LeapYearChecker {  
    public static void main(String[] args) {  
        int year = 2024; // Hardcoded year  
  
        // Check if the year is a leap year  
        if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {  
            System.out.println(year + " is a leap year.");  
        } else {  
            System.out.println(year + " is not a leap year.");  
        }  
    }  
}
```

Listing 57: Leap Year Checker Java Program

4. Run the Java program:
 - Click on the **Run** menu and select **Run Project** (or press F6).
 - The output window will display whether the year is a leap year or not.

17.4 Conclusion

The Java program successfully demonstrates the use of control flow statements (if-else) to determine whether a configured year is a leap year. This exercise highlights the importance of conditional statements in making decisions based on specific criteria in Java programming.

17.5 Precautions

- Ensure that Apache NetBeans IDE and JDK are properly installed and configured on your computer.
- Double-check the code for syntax errors before running the program.
- Save your work frequently to avoid losing any changes.
- Use meaningful variable names to enhance code readability.

18 Create a Java Program to Demonstrate switch-case Statement

18.1 Aim

To create a Java program that output Day based on a configured number using switch-case statement.

18.2 Requirements

- Apache NetBeans IDE installed on your computer.
- Java Development Kit (JDK) installed on your computer.
- Basic knowledge of Java programming.

18.3 Procedure

1. Open Apache NetBeans IDE on your computer.
2. Create a new Java project:
 - Click on **File** → **New Project**.
 - In the New Project dialog, select **Java** from the Categories list and **Java Application** from the Projects list. Click **Next**.
 - Enter a project name (e.g., "DayOfWeekFinder") and choose a location to save the project. Ensure that the "Create Main Class" option is checked. Click **Finish**.
3. Write the Java program:
 - In the Project Explorer, expand the project folder and locate the main class file (e.g., DayOfWeekFinder.java).
 - Double-click on the main class file to open it in the editor.
 - Replace the existing code with the following code:

```
public class DayOfWeekFinder {
    public static void main(String[] args) {
        int dayNumber = 3; // Hardcoded number for demonstration
        String dayName;

        switch (dayNumber) {
            case 1: dayName = "Monday"; break;
            case 2: dayName = "Tuesday"; break;
            case 3: dayName = "Wednesday"; break;
            case 4: dayName = "Thursday"; break;
            case 5: dayName = "Friday"; break;
            case 6: dayName = "Saturday"; break;
            case 7: dayName = "Sunday"; break;
            default: dayName = "Invalid input!";
        }

        System.out.println("The day is: " + dayName);
    }
}
```

Listing 58: Day of Week Finder Java Program

4. Run the Java program:
 - Click on the **Run** menu and select **Run Project** (or press F6).
 - The output window will display the corresponding day of the week or an invalid input message.

18.4 Conclusion

The Java program successfully demonstrates the use of the switch-case statement to map numbers to days of the week. This exercise highlights the importance of control flow statements in handling multiple conditions in Java programming.

18.5 Precautions

- Ensure that Apache NetBeans IDE and JDK are properly installed and configured on your computer.
- Double-check the code for syntax errors before running the program.
- Save your work frequently to avoid losing any changes.
- Use meaningful variable names to enhance code readability.

19 Create a Java Program to Demonstrate while Loop

19.1 Aim

To create a Java program that prints the first 10 natural numbers using a while loop.

19.2 Requirements

- Apache NetBeans IDE installed on your computer.
- Java Development Kit (JDK) installed on your computer.
- Basic knowledge of Java programming.

19.3 Procedure

1. Open Apache NetBeans IDE on your computer.
2. Create a new Java project:
 - Click on **File** → **New Project**.
 - In the New Project dialog, select **Java** from the Categories list and **Java Application** from the Projects list. Click **Next**.
 - Enter a project name (e.g., "NaturalNumbersPrinter") and choose a location to save the project. Ensure that the "Create Main Class" option is checked. Click **Finish**.
3. Write the Java program:
 - In the Project Explorer, expand the project folder and locate the main class file (e.g., NaturalNumbersPrinter.java).
 - Double-click on the main class file to open it in the editor.
 - Replace the existing code with the following code:

```
public class NaturalNumbersPrinter {  
    public static void main(String[] args) {  
        int number = 1; // Starting number  
  
        System.out.println("First 10 natural numbers:");  
  
        // Using while loop to print first 10 natural numbers  
        while (number <= 10) {  
            System.out.println(number);  
            number++; // Increment the number by 1  
        }  
    }  
}
```

Listing 59: Natural Numbers Printer Java Program

4. Run the Java program:
 - Click on the **Run** menu and select **Run Project** (or press F6).
 - The output window will display the first 10 natural numbers.

19.4 Conclusion

The Java program successfully demonstrates the use of a while loop to print the first 10 natural numbers. This exercise highlights the importance of loops in executing repetitive tasks in Java programming.

19.5 Precautions

- Ensure that Apache NetBeans IDE and JDK are properly installed and configured on your computer.
- Double-check the code for syntax errors before running the program.
- Save your work frequently to avoid losing any changes.
- Use meaningful variable names to enhance code readability.

20 Create a Java Program to Demonstrate for Loop and Arrays

20.1 Aim

To create a Java program to store a list of marks in an array and display them.

20.2 Requirements

- Apache NetBeans IDE installed on your computer.
- Java Development Kit (JDK) installed on your computer.
- Basic knowledge of Java programming.

20.3 Procedure

1. Open Apache NetBeans IDE on your computer.
2. Create a new Java project:
 - Click on **File** → **New Project**.
 - In the New Project dialog, select **Java** from the Categories list and **Java Application** from the Projects list. Click **Next**.
 - Enter a project name (e.g., "MarksList") and choose a location to save the project. Ensure that the "Create Main Class" option is checked. Click **Finish**.
3. Write the Java program:
 - In the Project Explorer, expand the project folder and locate the main class file (e.g., MarksList.java).
 - Double-click on the main class file to open it in the editor.
 - Replace the existing code with the following code:

```
public class MarksList {
    public static void main(String[] args) {
        // Declare and initialize an array to store marks
        int[] marks = {85, 90, 78, 92, 88};

        System.out.println("List of Marks:");

        // Use a for loop to iterate through the array and print each mark
        for (int i = 0; i < marks.length; i++) {
            System.out.println("Student " + (i + 1) + ": " + marks[i] + " marks");
        }
    }
}
```

Listing 60: Marks List Java Program

4. Run the Java program:
 - Click on the **Run** menu and select **Run Project** (or press F6).
 - The output window will display the list of marks for each student.

20.4 Conclusion

The Java program successfully demonstrates the use of arrays to store and display a list of marks. This exercise highlights the importance of arrays in managing collections of data in Java programming.

20.5 Precautions

- Ensure that Apache NetBeans IDE and JDK are properly installed and configured on your computer.
- Double-check the code for syntax errors before running the program.
- Save your work frequently to avoid losing any changes.
- Use meaningful variable names to enhance code readability.

21 Create a Java Program to Demonstrate Methods

21.1 Aim

To create a Java program that demonstrates the use of methods by calculating the area of a rectangle.

21.2 Requirements

- Apache NetBeans IDE installed on your computer.
- Java Development Kit (JDK) installed on your computer.
- Basic knowledge of Java programming.

21.3 Procedure

1. Open Apache NetBeans IDE on your computer.
2. Create a new Java project:
 - Click on **File** → **New Project**.
 - In the New Project dialog, select **Java** from the Categories list and **Java Application** from the Projects list. Click **Next**.
 - Enter a project name (e.g., "AreaCalculator") and choose a location to save the project. Ensure that the "Create Main Class" option is checked. Click **Finish**.
3. Write the Java program:
 - In the Project Explorer, expand the project folder and locate the main class file (e.g., AreaCalculator.java).
 - Double-click on the main class file to open it in the editor.
 - Replace the existing code with the following code:

```
public class AreaCalculator {  
    public static double calculateArea(double length, double width) {  
        return length * width;  
    }  
  
    public static void main(String[] args) {  
        System.out.println("The area is: " + calculateArea(5.0, 3.0));  
    }  
}
```

Listing 61: Area Calculator Java Program

4. Run the Java program:
 - Click on the **Run** menu and select **Run Project** (or press F6).
 - The output window will display the area of the rectangle.

21.4 Conclusion

The Java program successfully demonstrates the use of methods to calculate the area of a rectangle. This exercise highlights the importance of methods in organizing code and performing specific tasks in Java programming.

21.5 Precautions

- Ensure that Apache NetBeans IDE and JDK are properly installed and configured on your computer.
- Double-check the code for syntax errors before running the program.
- Save your work frequently to avoid losing any changes.
- Use meaningful method and variable names to enhance code readability.

22 Create a Java Program to Demonstrate OOP

22.1 Aim

To create a Java program that demonstrates Object-Oriented Programming (OOP) concepts by defining a class for a Car and creating objects with constructors and displaying car details.

22.2 Requirements

- Apache NetBeans IDE installed on your computer.
- Java Development Kit (JDK) installed on your computer.
- Basic knowledge of Java programming and OOP concepts.

22.3 Procedure

1. Open Apache NetBeans IDE on your computer.
2. Create a new Java project:
 - Click on **File** → **New Project**.
 - In the New Project dialog, select **Java** from the Categories list and **Java Application** from the Projects list. Click **Next**.
 - Enter a project name (e.g., "CarDetails") and choose a location to save the project. Ensure that the "Create Main Class" option is checked. Click **Finish**.
3. Write the Java program:
 - In the Project Explorer, expand the project folder and locate the main class file (e.g., CarDetails.java).
 - Double-click on the main class file to open it in the editor.
 - Replace the existing code with the following code:

```
class Car {
    String make;
    int year;

    public Car(String make, int year) {
        this.make = make;
        this.year = year;
    }

    void displayDetails() {
        System.out.println("Make: " + make + ", Year: " + year);
    }
}

public class CarDetails {
    public static void main(String[] args) {
        Car car1 = new Car("Toyota", 2020);
        Car car2 = new Car("Honda", 2021);

        car1.displayDetails();
        car2.displayDetails();
    }
}
```

Listing 62: Car Details Java Program

4. Run the Java program:
 - Click on the **Run** menu and select **Run Project** (or press F6).
 - The output window will display the details of both car objects.

22.4 Conclusion

The Java program successfully demonstrates Object-Oriented Programming (OOP) concepts by defining a Car class and creating objects to display car details. This exercise highlights the importance of classes and objects in organizing code and modeling real-world entities in Java programming.

22.5 Precautions

- Ensure that Apache NetBeans IDE and JDK are properly installed and configured on your computer.
- Double-check the code for syntax errors before running the program.
- Save your work frequently to avoid losing any changes.
- Use meaningful class, method, and variable names to enhance code readability.

23 Create a Java Program to Demonstrate Access Modifiers and GettersSetters

23.1 Aim

To create a Java program that demonstrates the use of access modifiers and getter/setter methods by defining a class for a Student with private attributes.

23.2 Requirements

- Apache NetBeans IDE installed on your computer.
- Java Development Kit (JDK) installed on your computer.
- Basic knowledge of Java programming and OOP concepts.

23.3 Procedure

1. Open Apache NetBeans IDE on your computer.
2. Create a new Java project:
 - Click on **File** → **New Project**.
 - In the New Project dialog, select **Java** from the Categories list and **Java Application** from the Projects list. Click **Next**.
 - Enter a project name (e.g., "StudentDetails") and choose a location to save the project. Ensure that the "Create Main Class" option is checked. Click **Finish**.
3. Write the Java program:
 - In the Project Explorer, expand the project folder and locate the main class file (e.g., StudentDetails.java).
 - Double-click on the main class file to open it in the editor.
 - Replace the existing code with the following code:

```
class Student {
    private String name;
    private int age;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}

public class StudentDetails {
    public static void main(String[] args) {
        Student student = new Student();
        student.setName("John Doe");
        student.setAge(20);

        System.out.println("Name: " + student.getName());
        System.out.println("Age: " + student.getAge());
    }
}
```

```
}
```

Listing 63: Student Details Java Program

4. Run the Java program:

- Click on the **Run** menu and select **Run Project** (or press F6).
- The output window will display the details of the student object.

23.4 Conclusion

The Java program successfully demonstrates the use of access modifiers and getter/setter methods by defining a Student class with private attributes. This exercise highlights the importance of encapsulation in protecting data and providing controlled access in Java programming.

23.5 Precautions

- Ensure that Apache NetBeans IDE and JDK are properly installed and configured on your computer.
- Double-check the code for syntax errors before running the program.
- Save your work frequently to avoid losing any changes.
- Use meaningful class, method, and variable names to enhance code readability.

24 Create a Java Program to Demonstrate Data Input using Scanner Class

24.1 Aim

To create a Java program that demonstrates data input using the Scanner class by reading user input for a student's name, age, and grade.

24.2 Requirements

- Apache NetBeans IDE installed on your computer.
- Java Development Kit (JDK) installed on your computer.
- Basic knowledge of Java programming.

24.3 Procedure

1. Open Apache NetBeans IDE on your computer.
2. Create a new Java project:
 - Click on **File** → **New Project**.
 - In the New Project dialog, select **Java** from the Categories list and **Java Application** from the Projects list. Click **Next**.
 - Enter a project name (e.g., "StudentInput") and choose a location to save the project. Ensure that the "Create Main Class" option is checked. Click **Finish**.
3. Write the Java program:
 - In the Project Explorer, expand the project folder and locate the main class file (e.g., StudentInput.java).
 - Double-click on the main class file to open it in the editor.
 - Replace the existing code with the following code:

```
import java.util.Scanner;
public class StudentInput {
    public static void main(String[] args) {
        // Create a Scanner object to read input
        Scanner scanner = new Scanner(System.in);

        // Prompt user for student details
        System.out.print("Enter student's name: ");
        String name = scanner.nextLine();

        System.out.print("Enter student's age: ");
        int age = scanner.nextInt();

        System.out.print("Enter student's grade: ");
        String grade = scanner.next();

        // Display the entered student details
        System.out.println("\nStudent Details:");
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Grade: " + grade);

        // Close the scanner
        scanner.close();
    }
}
```

Listing 64: Student Input Java Program

4. Run the Java program:

- Click on the **Run** menu and select **Run Project** (or press F6).
- Enter the student's name, age, and grade when prompted in the output window and press Enter after each input.
- The output window will display the entered student details.

24.4 Conclusion

The Java program successfully demonstrates data input using the Scanner class by reading user input for a student's name, age, and grade. This exercise highlights the importance of user input handling in Java programming.

24.5 Precautions

- Ensure that Apache NetBeans IDE and JDK are properly installed and configured on your computer.
 - Double-check the code for syntax errors before running the program.
 - Save your work frequently to avoid losing any changes.
 - Use meaningful variable names to enhance code readability.
-

25 Create a Java Program to Demonstrate Arrays and String Manipulation

25.1 Aim

To create a Java program that demonstrates the use of arrays and string manipulation by storing a list of names and performing various operations on them.

25.2 Requirements

- Apache NetBeans IDE installed on your computer.
- Java Development Kit (JDK) installed on your computer.
- Basic knowledge of Java programming.

25.3 Procedure

1. Open Apache NetBeans IDE on your computer.
2. Create a new Java project:
 - Click on **File** → **New Project**.
 - In the New Project dialog, select **Java** from the Categories list and **Java Application** from the Projects list. Click **Next**.
 - Enter a project name (e.g., "NameList") and choose a location to save the project. Ensure that the "Create Main Class" option is checked. Click **Finish**.
3. Write the Java program:
 - In the Project Explorer, expand the project folder and locate the main class file (e.g., NameList.java).
 - Double-click on the main class file to open it in the editor.
 - Replace the existing code with the following code:

```
public class NameList {
    public static void main(String[] args) {
        // Declare and initialize an array to store names
        String[] names = {"Alice", "Bob", "Charlie", "Diana", "Ethan"};

        System.out.println("List of Names:");

        // Use a for loop to iterate through the array and print each name
        for (int i = 0; i < names.length; i++) {
            System.out.println("Name " + (i + 1) + ": " + names[i]);
        }

        // Perform string manipulation: Convert names to uppercase
        System.out.println("\nNames in Uppercase:");
        for (String name : names) {
            System.out.println(name.toUpperCase());
        }
    }
}
```

Listing 65: Name List Java Program

4. Run the Java program:
 - Click on the **Run** menu and select **Run Project** (or press F6).
 - The output window will display the list of names and their uppercase versions.

25.4 Conclusion

The Java program successfully demonstrates the use of arrays and string manipulation by storing a list of names and converting them to uppercase. This exercise highlights the importance of arrays for data storage and string methods for manipulating text in Java programming.

25.5 Precautions

- Ensure that Apache NetBeans IDE and JDK are properly installed and configured on your computer.
 - Double-check the code for syntax errors before running the program.
 - Save your work frequently to avoid losing any changes.
 - Use meaningful variable names to enhance code readability.
-

26 Create a Java Program to Demonstrate Exception Handling

26.1 Aim

To create a Java program that demonstrates exception handling by handling division by zero errors.

26.2 Requirements

- Apache NetBeans IDE installed on your computer.
- Java Development Kit (JDK) installed on your computer.
- Basic knowledge of Java programming.

26.3 Procedure

1. Open Apache NetBeans IDE on your computer.
2. Create a new Java project:
 - Click on **File** → **New Project**.
 - In the New Project dialog, select **Java** from the Categories list and **Java Application** from the Projects list. Click **Next**.
 - Enter a project name (e.g., "ExceptionHandlingDivision") and choose a location to save the project. Ensure that the "Create Main Class" option is checked. Click **Finish**.
3. Write the Java program:
 - In the Project Explorer, expand the project folder and locate the main class file (e.g., ExceptionHandlingDivision.java).
 - Double-click on the main class file to open it in the editor.
 - Replace the existing code with the following code:

```
import java.util.Scanner;
public class ExceptionHandlingDivision {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter numerator: ");
        int numerator = scanner.nextInt();

        System.out.print("Enter denominator: ");
        int denominator = scanner.nextInt();

        try {
            int result = numerator / denominator;
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Error: Division by zero is not allowed.");
        } finally {
            scanner.close();
        }
    }
}
```

Listing 66: Exception Handling Division Java Program

4. Run the Java program:
 - Click on the **Run** menu and select **Run Project** (or press F6).
 - Enter the numerator and denominator when prompted in the output window and press Enter after each input.
 - The output window will display the result or an error message if division by zero is attempted.

26.4 Conclusion

The Java program successfully demonstrates exception handling by managing division by zero errors. This exercise highlights the importance of using try-catch blocks to handle runtime exceptions and ensure program stability in Java programming.

26.5 Precautions

- Ensure that Apache NetBeans IDE and JDK are properly installed and configured on your computer.
- Double-check the code for syntax errors before running the program.
- Save your work frequently to avoid losing any changes.
- Use meaningful variable names to enhance code readability.

27 Create a Java Program to Demonstrate File Database Connectivity using JDBC

27.1 Aim

To create a Java program that demonstrates file database connectivity using JDBC by connecting to a MySQL database and performing basic operations.

27.2 Requirements

- Apache NetBeans IDE installed on your computer.
- Java Development Kit (JDK) installed on your computer.
- MySQL database server installed and running on your computer.
- MySQL Connector/J JDBC driver added to your project libraries.
- Basic knowledge of Java programming and SQL.

27.3 Procedure

1. Open Apache NetBeans IDE on your computer.
2. Create a new Java project:
 - Click on **File** → **New Project**.
 - In the New Project dialog, select **Java** from the Categories list and **Java Application** from the Projects list. Click **Next**.
 - Enter a project name (e.g., "JDBCSchool") and choose a location to save the project. Ensure that the "Create Main Class" option is checked. Click **Finish**.
3. Add MySQL Connector/J JDBC driver to the project:
 - Right-click on the project name in the Project Explorer and select **Properties**.
 - In the Properties dialog, select **Libraries** from the Categories list.
 - Click on the **Add JAR/Folder** button and navigate to the location where you downloaded the MySQL Connector/J JDBC driver JAR file. Select the JAR file and click **Open**.
 - Click **OK** to close the Properties dialog.
4. Write the Java program:
 - In the Project Explorer, expand the project folder and locate the main class file (e.g., JDBC-School.java).
 - Double-click on the main class file to open it in the editor.
 - Replace the existing code with the following code:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
public class JDBCSchool {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/School"; // to connect to 'School
        ' database
        String user = "your_username"; // Replace with your database username
        String password = "your_password"; // Replace with your database password
        try {
            // Establishing a connection
            Connection connection = DriverManager.getConnection(url, user, password
        );
        System.out.println("Connected to the database.");
    }
}
```

```

        // Creating a statement
        Statement statement = connection.createStatement();

        // Executing a query to fetch records from Teacher table
        String query = "SELECT Teacher_ID, First_Name, Last_Name, Salary FROM
Teacher LIMIT 5";
        ResultSet resultSet = statement.executeQuery(query);

        // Processing the result set
        System.out.println("\nTeacher Records:");
        while (resultSet.next()) {
            int id = resultSet.getInt("Teacher_ID");
            String firstName = resultSet.getString("First_Name");
            String lastName = resultSet.getString("Last_Name");
            double salary = resultSet.getDouble("Salary");
            System.out.println("ID: " + id + ", Name: " + firstName + " " +
lastName + ", Salary: " + salary);
        }

        // Closing the connection
        resultSet.close();
        statement.close();
        connection.close();
        System.out.println("\nDisconnected from the database.");
    } catch (Exception e) {
        System.out.println("Error: " + e.getMessage());
        e.printStackTrace();
    }
}
}
}

```

Listing 67: JDBC School Java Program

5. Run the Java program:

- Click on the **Run** menu and select **Run Project** (or press F6).
- The output window will display the data retrieved from the database table.

27.4 Conclusion

The Java program successfully demonstrates file database connectivity using JDBC by connecting to a MySQL database and performing basic operations. This exercise highlights the importance of JDBC in enabling Java applications to interact with databases.

27.5 Precautions

- Ensure that Apache NetBeans IDE, JDK, and MySQL database server are properly installed and configured on your computer.
- Double-check the database URL, username, password, and table/column names in the code before running the program.
- Save your work frequently to avoid losing any changes.
- Use meaningful variable names to enhance code readability.

28 Create a Java Program to Demonstrate Assertions

28.1 Aim

To create a Java program that demonstrates the use of assertions to validate conditions during runtime.

28.2 Requirements

- Apache NetBeans IDE installed on your computer.
- Java Development Kit (JDK) installed on your computer.
- Basic knowledge of Java programming.

28.3 Procedure

1. Open Apache NetBeans IDE on your computer.
2. Create a new Java project:
 - Click on **File** → **New Project**.
 - In the New Project dialog, select **Java** from the Categories list and **Java Application** from the Projects list. Click **Next**.
 - Enter a project name (e.g., "AssertionDemo") and choose a location to save the project. Ensure that the "Create Main Class" option is checked. Click **Finish**.
3. Write the Java program:
 - In the Project Explorer, expand the project folder and locate the main class file (e.g., AssertionDemo.java).
 - Double-click on the main class file to open it in the editor.
 - Replace the existing code with the following code:

```
public class AssertionDemo {
    public static void main(String[] args) {
        int value = 10;

        // Assertion to check if value is positive
        assert value > 0 : "Value must be positive";

        System.out.println("Value is: " + value);

        // Assertion to check if value is less than 20
        assert value < 20 : "Value must be less than 20";

        System.out.println("All assertions passed.");
    }
}
```

Listing 68: Assertion Demo Java Program

4. Run the Java program with assertions enabled:
 - Click on the **Run** menu and select **Set Project Configuration** → **Customize....**
 - In the Project Properties dialog, select **Run** from the Categories list.
 - In the VM Options field, add the following option to enable assertions: **-ea**
 - Click **OK** to close the Project Properties dialog.
 - Now, click on the **Run** menu and select **Run Project** (or press F6).
 - The output window will display the value and confirm that all assertions passed.

28.4 Conclusion

The Java program successfully demonstrates the use of assertions to validate conditions during runtime. This exercise highlights the importance of assertions in identifying logical errors and ensuring program correctness in Java programming.

28.5 Precautions

- Ensure that Apache NetBeans IDE and JDK are properly installed and configured on your computer.
- Double-check the code for syntax errors before running the program.
- Save your work frequently to avoid losing any changes.
- Use meaningful variable names to enhance code readability.

29 Create a Java Program to Demonstrate Multithreading

29.1 Aim

To create a Java program that demonstrates multithreading by creating and running multiple threads concurrently.

29.2 Requirements

- Apache NetBeans IDE installed on your computer.
- Java Development Kit (JDK) installed on your computer.
- Basic knowledge of Java programming and multithreading concepts.

29.3 Procedure

1. Open Apache NetBeans IDE on your computer.
2. Create a new Java project:
 - Click on **File** → **New Project**.
 - In the New Project dialog, select **Java** from the Categories list and **Java Application** from the Projects list. Click **Next**.
 - Enter a project name (e.g., "MultithreadingDemo") and choose a location to save the project. Ensure that the "Create Main Class" option is checked. Click **Finish**.
3. Write the Java program:
 - In the Project Explorer, expand the project folder and locate the main class file (e.g., MultithreadingDemo.java).
 - Double-click on the main class file to open it in the editor.
 - Replace the existing code with the following code:

```
class MyThread extends Thread {
    private String threadName;

    MyThread(String name) {
        threadName = name;
    }

    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(threadName + " - Count: " + i);
            try {
                Thread.sleep(500); // Sleep for 500 milliseconds
            } catch (InterruptedException e) {
                System.out.println(threadName + " interrupted.");
            }
        }
        System.out.println(threadName + " finished execution.");
    }
}

public class MultithreadingDemo {
    public static void main(String[] args) {
        MyThread thread1 = new MyThread("Thread 1");
        MyThread thread2 = new MyThread("Thread 2");

        // Start the threads
        thread1.start();
        thread2.start();
    }
}
```

Listing 69: Multithreading Demo Java Program

4. Run the Java program:

- Click on the **Run** menu and select **Run Project** (or press F6).
- The output window will display the concurrent execution of both threads.

29.4 Conclusion

The Java program successfully demonstrates multithreading by creating and running multiple threads concurrently. This exercise highlights the importance of multithreading in improving application performance and responsiveness in Java programming.

29.5 Precautions

- Ensure that Apache NetBeans IDE and JDK are properly installed and configured on your computer.
 - Double-check the code for syntax errors before running the program.
 - Save your work frequently to avoid losing any changes.
 - Use meaningful class and variable names to enhance code readability.
-

30 Create a Java Program to Demonstrate Wrapper Classes

30.1 Aim

To create a Java program that demonstrates the use of wrapper classes by converting a primitive data type to its corresponding wrapper object and vice versa.

30.2 Requirements

- Apache NetBeans IDE installed on your computer.
- Java Development Kit (JDK) installed on your computer.
- Basic knowledge of Java programming.

30.3 Procedure

1. Open Apache NetBeans IDE on your computer.
2. Create a new Java project:
 - Click on **File** → **New Project**.
 - In the New Project dialog, select **Java** from the Categories list and **Java Application** from the Projects list. Click **Next**.
 - Enter a project name (e.g., "WrapperClassDemo") and choose a location to save the project. Ensure that the "Create Main Class" option is checked. Click **Finish**.
3. Write the Java program:
 - In the Project Explorer, expand the project folder and locate the main class file (e.g., WrapperClassDemo.java).
 - Double-click on the main class file to open it in the editor.
 - Replace the existing code with the following code:

```
public class WrapperClassDemo {
    public static void main(String[] args) {
        int primitiveInt = 10;

        // Explicit Boxing
        Integer wrapperInt = Integer.valueOf(primitiveInt);
        System.out.println("Wrapper Integer: " + wrapperInt);

        // Explicit Unboxing
        int unboxedInt = wrapperInt.intValue();
        System.out.println("Unboxed Integer: " + unboxedInt);

        // Automatic Boxing
        Integer autoBoxInt = 50;
        System.out.println("Auto-boxed Integer: " + autoBoxInt);

        // Automatic Unboxing
        int autoUnboxInt = autoBoxInt;
        System.out.println("Auto-unboxed Integer: " + autoUnboxInt);
    }
}
```

Listing 70: Wrapper Class Demo Java Program

4. Run the Java program:
 - Click on the **Run** menu and select **Run Project** (or press F6).
 - The output window will display the wrapper objects and their unboxed primitive values.

30.4 Conclusion

The Java program successfully demonstrates the use of wrapper classes by converting a primitive data type to its corresponding wrapper object and vice versa. This exercise highlights the importance of wrapper classes in providing object representation for primitive data types in Java programming.

30.5 Precautions

- Ensure that Apache NetBeans IDE and JDK are properly installed and configured on your computer.
- Double-check the code for syntax errors before running the program.
- Save your work frequently to avoid losing any changes.
- Use meaningful variable names to enhance code readability.