

**Godavari Foundations's**  
**Godavari College Of Engineering,**  
**Jalgaon**



**DEPARTMENT OF**  
**COMPUTER ENGINEERING**

**V SEMESTER**

**LAB MANUAL DBMS LABORATORY**

**SUBJECT CODE: BTCOL507**

**SESSION: August 2020 – December 2020**

**Faculty : Prof. MADHURI ZAWAR**

Godavari Foundation's  
Godavari College of Engineering, Jalgaon  
Department of Computer  
**Lab Manual**  
Database System Laboratory  
**Practical No :- 1**

**Date:-** 08-09-2020

**Name of Student:-** Shweta Ravindra Patil.

**Class:-** T.Y. Computer Engineering.

**Roll No:-** 34

**Title:** Defining schema for applications.

**Aim:** - Defining schema for applications such as Hospital Management System, Library Management System, Railway Reservation System, Banking System.

**Software Requirement :-** No Requirements.

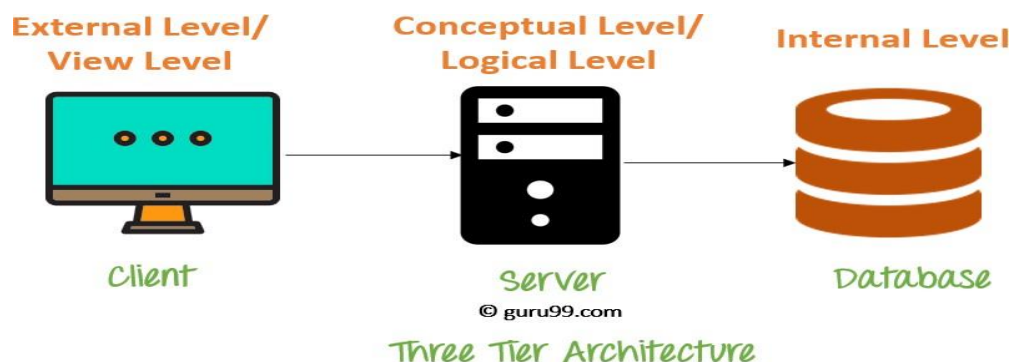
**Hardware Requirement :-** No Requirements.

**Theory:-**

**DBMS Schemas: Internal, Conceptual, External**

There are mainly three levels of data abstraction:

1. Internal Level: Actual PHYSICAL storage structure and access paths.
2. Conceptual or Logical Level: Structure and constraints for the entire database.
3. External or View level: Describes various user views



## **Internal Level/Schema**

The internal schema defines the physical storage structure of the database. The internal schema is a very low-level representation of the entire database. It contains multiple occurrences of multiple types of internal record. In the ANSI term, it is also called "stored record".

Facts about Internal schema:

- The internal schema is the lowest level of data abstraction
- It helps you to keep information about the actual representation of the entire database. Like the actual storage of the data on the disk in the form of records
- The internal view tells us what data is stored in the database and how
- It never deals with the physical devices. Instead, internal schema views a physical device as a collection of physical pages

## **Conceptual Schema/Level**

The conceptual schema describes the Database structure of the whole database for the community of users. This schema hides information about the physical storage structures and focuses on describing data types, entities, relationships, etc.

This logical level comes between the user level and physical storage view. However, there is only single conceptual view of a single database.

Facts about Conceptual schema:

- Defines all database entities, their attributes, and their relationships
- Security and integrity information
- In the conceptual level, the data available to a user must be contained in or derivable from the physical level

## **External Schema/Level**

An external schema describes the part of the database which specific user is interested in. It hides the unrelated details of the database from the user. There may be "n" number of external views for each database.

Each external view is defined using an external schema, which consists of definitions of various types of external record of that specific view.

An external view is just the content of the database as it is seen by some specific particular user. For example, a user from the sales department will see only sales related data.

Facts about external schema:

- An external level is only related to the data which is viewed by specific end users.
- This level includes some external schemas.
- External schema level is nearest to the user
- The external schema describes the segment of the database which is needed for a certain user group and hides the remaining details from the database from the specific user group

## **Goal of 3 level/schema of Database**

Objectives of using Three schema Architecture:

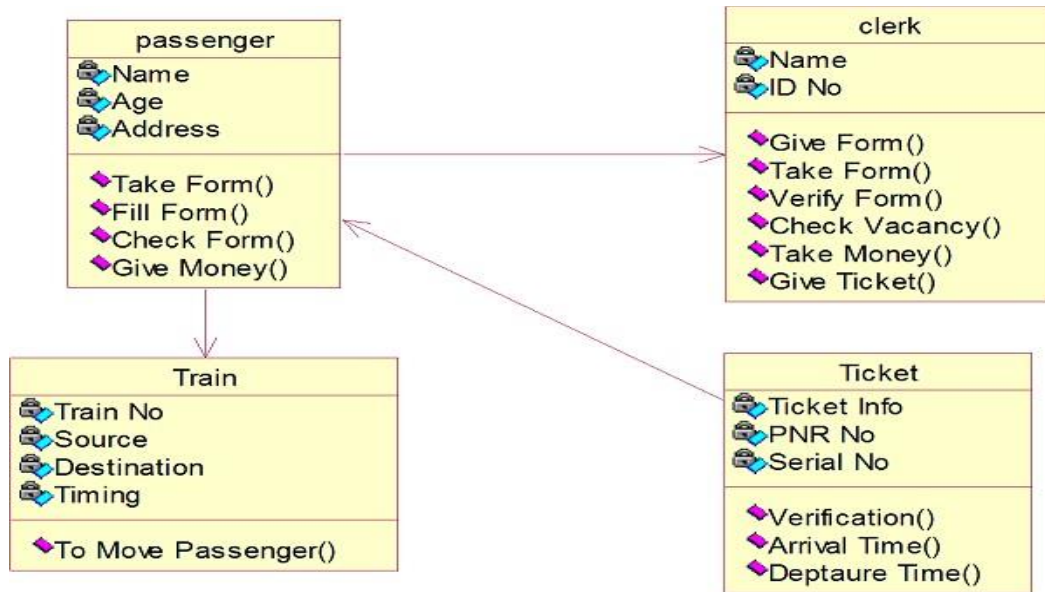
- Every user should be able to access the same data but able to see a customized view of the data.
- The user need not to deal directly with physical database storage detail.
- The DBA should be able to change the database storage structure without disturbing the user's views
- The internal structure of the database should remain unaffected when changes made to the physical aspects of storage.

## **Advantages Database Schema**

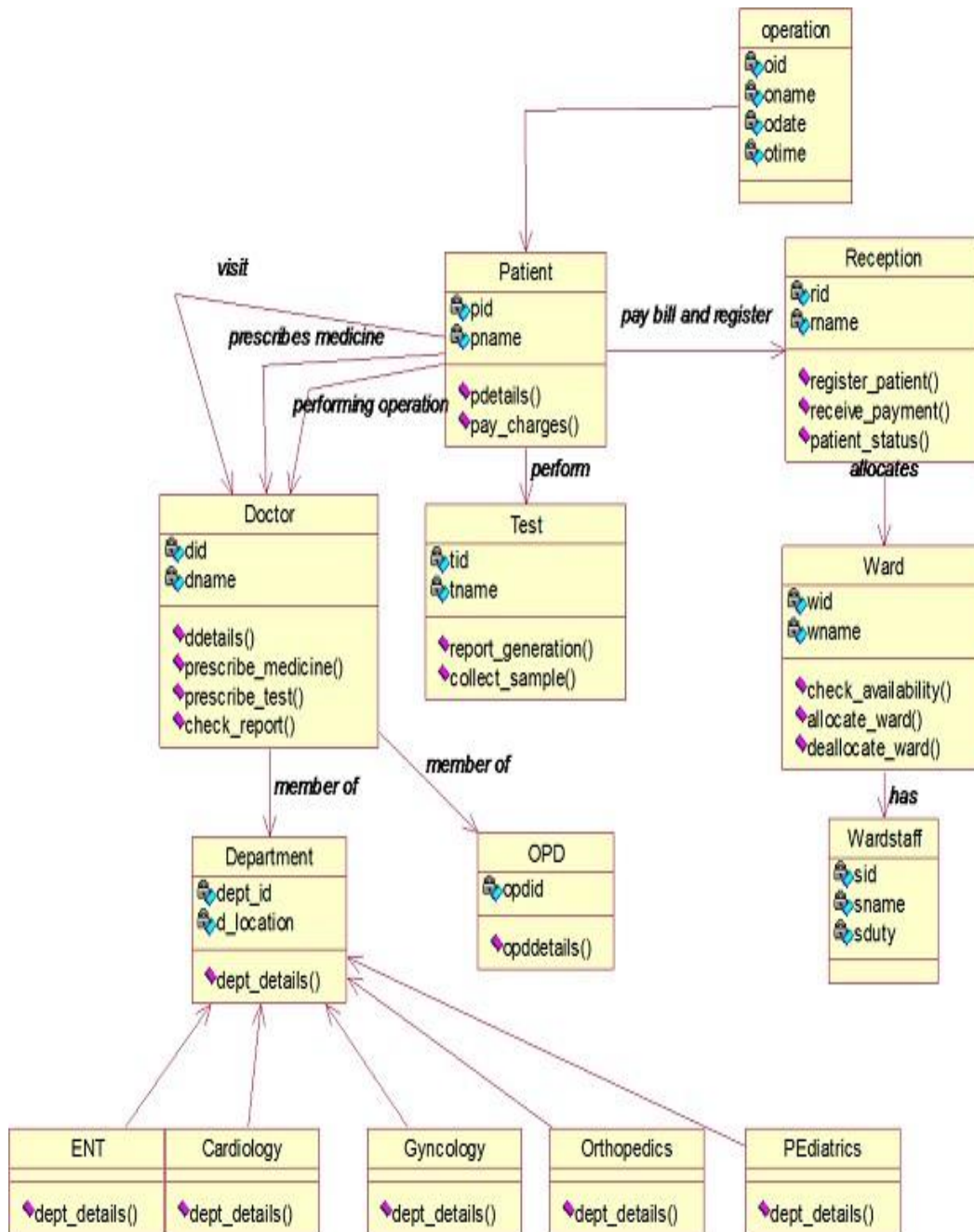
- You can manage data independent of the physical storage
- Faster Migration to new graphical environments
- DBMS Architecture allows you to make changes on the presentation level without affecting the other two layers
- As each tier is separate, it is possible to use different sets of developers
- It is more secure as the client doesn't have direct access to the database business logic
- In case of the failure of the one-tier no data loss as you are always secure by accessing the other tier

## **Disadvantages Database Schema**

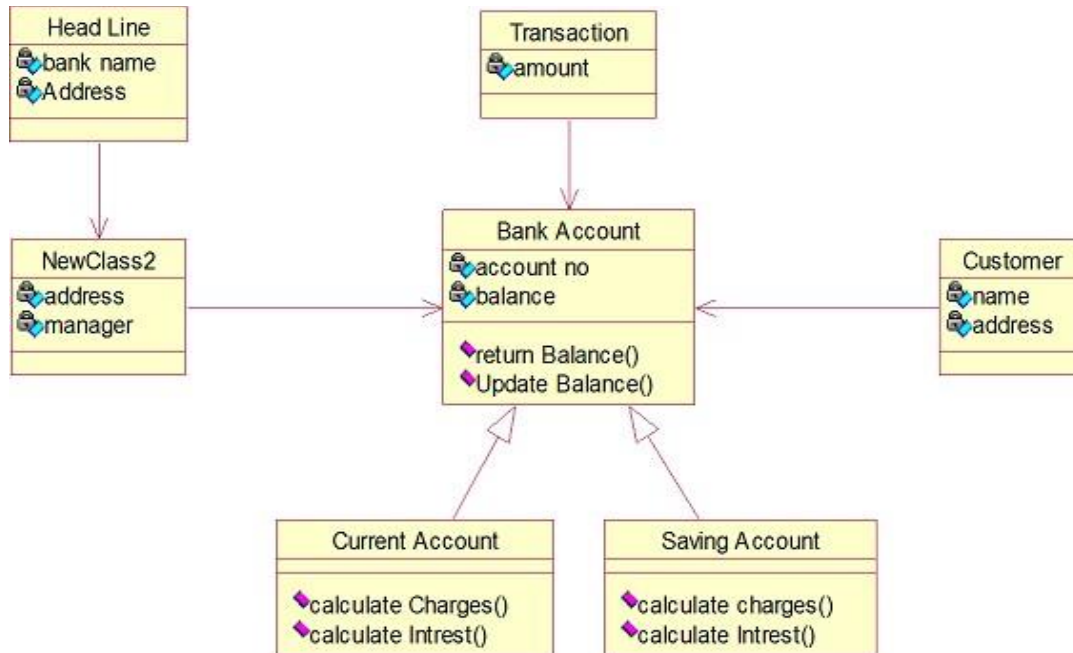
- Complete DB Schema is a complex structure which is difficult to understand for everyone
- Difficult to set up and maintain
- The physical separation of the tiers can affect the performance of the Database



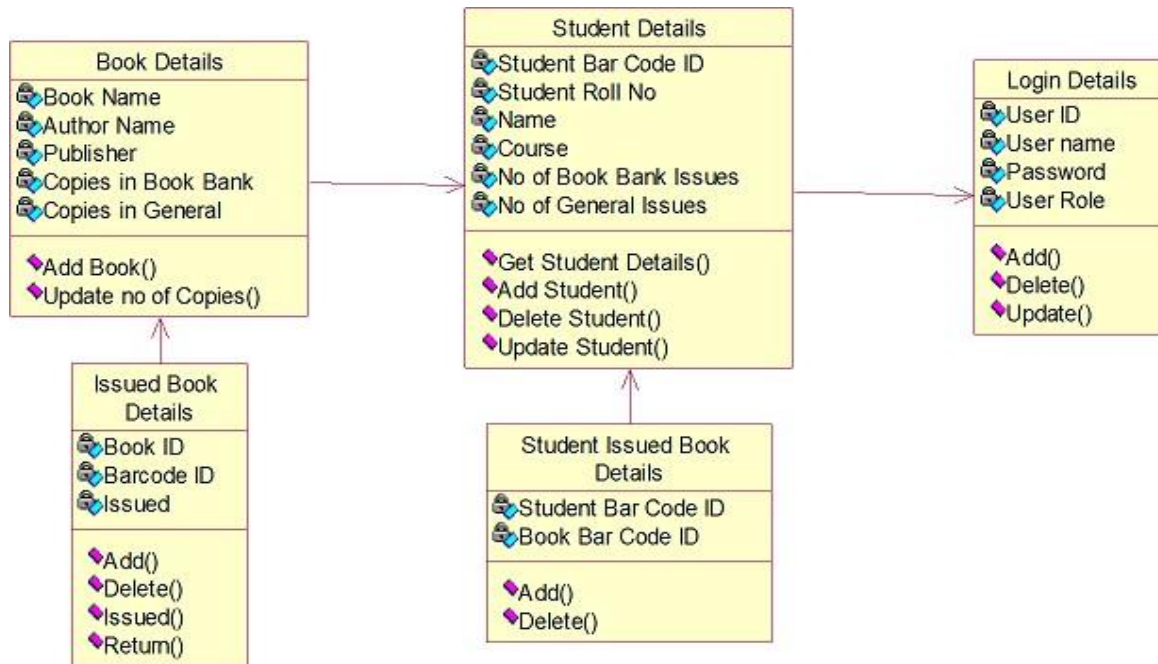
**Database Schema for Railway Reservation System**



**Database Schema for Hospital Management system**



## Database Schema for Banking System



## Database Schema for Library Management System

**Conclusion:-** In this Practical I learn Defining schema for applications such as Hospital Management System, Library Management System, Railway Reservation System, Banking System.



Godavari Foundation's  
Godavari College of Engineering, Jalgaon  
Department of Computer  
**Lab Manual**  
Database System Laboratory  
**Practical No:- 2**

**Date:-** 17-09-2020

**Name of Student:-** Shweta Ravindra Patil.

**Class:-** T.Y. Computer Engineering.

**Roll No:-** 34

**Title:** Creating tables, Renaming tables, Data constraints, Data insertion into a table.

**Aim:** - MYSQL Command to create table, rename table, Data constraints like (Primary key, Foreign key, Not Null), insert data into table.

**Software Requirement :-** MYSQL.

**Hardware Requirement :-** 2GB RAM.

**Theory:-**

**Create Database**

The CREATE DATABASE statement is used to create a new database.

**Syntax:-**

Create database <database\_name>

**Use Database**

**Syntax:-**

Use <database\_name>

**Create Table**

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

**Syntax:-**

```
Create <table_name>
( Column_name1 datatype(size),
.....
Column_nameN datatype(size));
```

**Example:-**

```
CREATE TABLE Persons (
PersonID int,
LastName   varchar(255),
FirstName  varchar(255),
Address   varchar(255), City
varchar(255));
```

**Rename Table**

RENAME TABLE syntax is used to change the name of a table.

**Syntax:-**

```
ALTER TABLE table_name RENAME TO new_table_name;
```

**Example:-**

```
ALTER TABLE STUDENTS RENAME TO ARTISTS;
```

**Data Constraints**

Constraints can be divided into the following two types,

- 1. Column level constraints:**Limits only column data.
- 2. Table level constraints:**Limits whole table data.

Constraints applied to a table are:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

**Primary Key Constraint**

Primary key constraint uniquely identifies each record in a database. A Primary Key must contain unique value and it must not contain null value.

### Using **PRIMARY KEY** constraint at Table Level

CREATE table Student (s\_id int PRIMARY KEY, Name varchar(60) NOT NULL, Age int); The above command will creates a PRIMARY KEY on the s\_id.

### Using **PRIMARY KEY** constraint at Column Level

ALTER table Student ADD PRIMARY KEY (s\_id);

The above command will creates a PRIMARY KEY on the s\_id.

### Foreign Key Constraint

FOREIGN KEY is used to relate two tables.

#### Customer\_Detail Table

c_id	Customer_Name	address
101	Adam	Noida
102	Alex	Delhi
103	Stuart	Rohtak

#### Order\_Detail Table

Order_id	Order_Name	c_id
10	Order1	101
11	Order2	103
12	Order3	102

In **Customer\_Detail** table, **c\_id** is the primary key which is set as foreign key in **Order\_Detail** table. The value that is entered in **c\_id** which is set as foreign key in **Order\_Detail** table must be present in **Customer\_Detail** table where it is set as primary key. This prevents invalid data to be inserted into **c\_id** column of **Order\_Detail** table.

### Using FOREIGN KEY constraint at Table Level

```
CREATE table Order_Detail ( order_id  
int PRIMARY KEY, order_name  
varchar(60) NOT NULL,  
c_id int FOREIGN KEY REFERENCES Customer_Detail(c_id));
```

### Using FOREIGN KEY constraint at Column Level

```
ALTER table Order_Detail ADD FOREIGN KEY (c_id) REFERENCES Customer_Detail(c_id);
```

#### NOT NULL Constraint :

**NOT NULL** constraint restricts a column from having a NULL value. Once NOT NULL constraint is applied to a column, you cannot pass a null value to that column.

#### Example:-

```
CREATE TABLE Student(s_id int NOT NULL, Name varchar(60), Age int);
```

### Create Constraints on the table

#### Column Constraints

##### Syntax:-

```
Create table <table_name>  
( Column_name1 datatype(size) constraint <constraint_name> Primary key,  
Column_name2 datatype(size) constraint <constraint_name> references  
referenced_table[(primary_column_name of referenced table)],  
Column_name3 datatype(size) constraint <constraint_name> Check(<condition>),  
Column_name4 datatype(size) NOT NULL);
```

#### Table Constraints

##### Syntax:-

```
Create table <table_name>  
(Column_name1 datatype(size),  
...  
...)
```

```
Column_name N datatype(size),  
Constraint <constraint_name> Primary key (column_name1),  
Constraint <constraint_name> Foreign key(Foreign_column_name) references  
referenced_table[(primary_column_name of referenced table)],  
Constraint <constraint_name> Check(<condition>);  
);
```

### **View Table Structure**

#### **Syntax:-**

```
Desc <table_name>
```

### **INSERT Statement**

The SQL INSERT statement is used to insert a single or multiple data in a table.

Two ways to insert Data:

- Without specifying column name
- By specifying column name

#### **4. Without specifying column name**

##### **Syntax:-**

```
INSERT INTO TABLE_NAME VALUES (value1, value2,value3,..... Value N);
```

##### **Example:-**

```
INSERT INTO EMPLOYEE VALUES (6, 'Marry', 'Canada', 600000, 48);
```

#### **5. By specifying column name**

##### **Syntax:-**

```
INSERT INTO TABLE_NAME[(col1,col2,col3, ..... col N)]
```

```
VALUES (value1, value2,value 3,.....Value N);
```

##### **Example:-**

```
INSERT INTO EMPLOYEE (EMP_ID, EMP_NAME, AGE) VALUES (7, 'Jack', 40);
```

#### **1. To View All Columns**

##### **Syntax:-**

```
select * from <table_name>
```

## **2. To View Selective Columns**

### **Syntax:-**

```
select column_name1, column_name2 from <table_name>
```

### **MYSOL QUERIES:-**

#### **Create Table Queries:-**

```
create database student; use
```

```
student;
```

```
create table Branch  
(Branch_Code Char(20),  
Add1 VarChar(30), Add2  
VarChar(30),  
City VarChar(20));
```

```
desc Branch;
```

#### **Insert Data & Display Data Queries:-**

```
use student;
```

```
select * from Branch;
```

```
insert into Branch values('SBI', 'Shiv Colony', 'Dadawadi', 'Jalgaon'); insert
```

```
into Branch(Branch_Code, Add1, Add2, City)  
values  
( 'Axis Bank', 'New Pandit Colony', 'Sharanpur Road', 'Nashik'),  
( 'ICICI', '349 Business Point', 'Andheri East', 'Mumbai'), ( 'HDFC',  
'Ruby House', 'Kharadi Magarpatta Bye Pass', 'Pune'), ( 'Yes Bank',  
'206 GPO Square', 'VIP Road Civil Lines', 'Nagpur'), ( 'Godavari Bank',  
'Anandidas Complex', 'Vazirabad', 'Nanded');
```

```
select * from Branch;
```

```
insert into Branch(Branch_Code, City) values('Godavari Laxmi Bank', 'Jalgaon'); select
```

```
* from Branch;
```

#### **Rename Table Query:-**

```
alter table Branch rename to Branch1;
```

```
select * from Branch1;
```

### **Primary Key Constraint Query:-**

use student;

```
create table Fare
(Route_Code Char(7),
Route_Desc VarChar(25) NOT NULL,
Origin VarChar(15) NOT NULL,
Destination VarChar(15) NOT NULL,
First_Fare int(5),
Bus_Fare int(5),
Eco_Fare int(5),
constraint Route_pk Primary Key(Route_Code),
constraint First_Fare_Zero check(First_Fare > 0),
constraint Bus_Fare_Greater_First check(Bus_Fare < First_Fare),
constraint Eco_Fare_Greater_Business check(Eco_Fare < Bus_Fare));
```

desc Fare;

### **Foreign Key Constraint Query:-**

```
create table Flight_Sch
(Flightno Char(4),
Airbusno Char(5),
Route_Code Char(7),
Deprt_time Char(5),
Journey_hrs Char(6),
Flight_Day1 int(1),
Flight_Day2 int(1),
constraint Flightno_pk Primary Key(Flightno),
constraint Airbusno_fk Foreign Key(Airbusno) references Airbus(Airbusno),
constraint Route_Code_fk Foreign Key(Route_Code) references Fare(Route_Code));
```

desc Airbus;

desc Fare;

desc Flight\_Sch;

**Conclusion:-** In this Practical I learn MYSQL Command to create table, rename table, Data constraints like (Primary key, Foreign key, Not Null), insert data into table.

Godavari Foundation's  
**Godavari College of Engineering, Jalgaon**  
**Department of Computer**  
**Lab Manual**  
**Database System Laboratory**  
**Practical No :- 3**

**Date:-** 30-09-2020

**Name of Student:-** Shweta Ravindra Patil.

**Class:-** T.Y. Computer Engineering.

**Roll No:-** 34

**Title :-** Grouping data, aggregate functions, Oracle functions (mathematical, character functions).

**Aim :-** MYSQL command for Group By Clause, Statistical Built in Aggregate Functions and String Functions.

**Software Requirement :-** MYSQL.

**Hardware Requirement :-** 2GB RAM.

**Theory :-**

**GroupBy**

GROUP BY is used to group values from a column

Syntax:-

```
SELECT column_name(s)
FROM table_name WHERE
condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```



```
mysql> SELECT * FROM employee_tbl;
```

```
+-----+-----+-----+-----+
| id | name | work_date | daily_typing_pages |
+-----+-----+-----+-----+
| 1 | John | 2007-01-24 | 250 |
+-----+-----+-----+-----+
| 2 | Ram | 2007-05-27 | 220 |
| 3 | Jack | 2007-05-06 | 170 |
| 3 | Jack | 2007-04-06 | 100 |
| 4 | Jill | 2007-04-06 | 220 |
| 5 | Zara | 2007-06-06 | 300 |
| 5 | Zara | 2007-02-06 | 350 |
+-----+-----+-----+-----+
+
7 rows in set (0.00 sec)
```

```
mysql> SELECT COUNT(*) FROM employee_tbl;
```

```
+-----+
| COUNT(*) |
+-----+
| 7 |
+-----+
```

By using aggregate functions in conjunction with a GROUP BY clause as follows

```
mysql> SELECT name, COUNT(*) FROM employee_tbl GROUP BY name;
```

```
+-----+-----+
| name | COUNT(*) |
+-----+-----+
| name | COUNT(*) |
| Jack | 2 |
| Jill | 2 |
| John | 1 |
| Ram | 1 |
| Zara | 2 |
| Zara | 2 |
+-----+-----+
5 rows in set (0.04 sec)
5 rows in set (0.04 sec)
```

## **Aggregate Functions**

- **MathematicalFunction**

### **1.Count()Function**

count() function returns the total number of values in the expression.

#### **Example:-**

```
Mysql> SELECT COUNT(name) FROM employee;
```

### **6. Sum()Function**

sum() function returns the total summed (non-NULL) value of an expression.

#### **Example:-**

```
Mysql> SELECT SUM(working_hours) AS "Total working hours" FROM employee;
```

### **7. AVG()Function**

AVG() function calculates the average of the values specified in the column.

#### **Example:-**

```
Mysql> SELECT AVG(working_hours) AS "Average working hours" FROM employee
```

### **8. MIN() Function**

MIN() function returns the minimum (lowest) value of the specified column.

#### **Example:-**

```
mysql> SELECT MIN(working_hours) AS Minimum_working_hours FROM employee;
```

### **9. MAX()Function**

MAX() function returns the maximum (highest) value of the specified column.

#### **Example:-**

```
Mysql> SELECT MAX(working_hours) AS Maximum_working_hours FROM employee;
```

- **String Functions**

**CHAR\_LENGTH():** Doesn't work for SQL Server. Use LEN() for SQL Server. This function is used to find the length of a word.

**Syntax:-**SELECT char\_length('Hello!');

**Output:-**6

**CHARACTER\_LENGTH():** Doesn't work for SQL Server. Use LEN() for SQL Server. This function is used to find the length of a line.

**Syntax:-**SELECT CHARACTER\_LENGTH('geeks for geeks');

**Output:-**15

**CONCAT():**This function is used to add two words or strings.

**Syntax:-**SELECT 'Geeks' || ' ' || 'forGeeks' FROM dual;

**Output:-**„GeeksforGeeks“

**INSTR():**This function is used to find the occurrence of an alphabet.

**Syntax:-**INSTR('geeks for geeks', 'e');

**Output:-**2 (the first occurrence of „e“)

**Syntax:-**INSTR('geeks for geeks', 'e', 1,2 );

**Output:-**3 (the second occurrence of „e“)

**LCASE():**This function is used to convert the given string into lower case.

**Syntax:-**LCASE ("GeeksFor Geeks To Learn");

**Output:-**geeksforgeeks to learn

**LENGTH():**This function is used to find the length of a word.

**Syntax:-**LENGTH('GeeksForGeeks');

**Output:-**13

**LOWER():**This function is used to convert the upper case string into lower case.

**Syntax:-**SELECT LOWER('GEEKSFORGEEKS.ORG');

**Output:-**geeksforgeeks.org

**LTRIM():**This function is used to cut the given sub string from the original string.

**Syntax:-**LTRIM('123123geeks', '123');

**Output:-**geeks

**REPEAT():**This function is used to write the given string again and again till the number of times mentioned.

**Syntax:-**SELECT REPEAT('geeks', 2);

**Output:-**geeksgeeks

**REPLACE():**This function is used to cut the given string by removing the given sub string.

**Syntax:-**REPLACE('123geeks123', '123');

**Output:-**geeks

**REVERSE():**This function is used to reverse a string.  
**Syntax:-**SELECT

REVERSE('geeksforgeeks.org');

**Output:-**„gro.skeegrofkskeeg“

**STRCMP():**This function is used to compare 2 strings.

**Syntax:-**SELECT STRCMP('google.com', 'geeksforgeeks.com');

**Output:-**-1

**UCASE():**This function is used to make the string in upper case.

**Syntax:-**UCASE ("GeeksForGeeks");

**Output:-**GEEKSFORGEEKS

## **MYSQL QUERIES**

### **Group By Queries:-**

use student;

select \* from Reservation;

select Branch\_Code, sum(Total\_Fare) TotalFare  
from Reservation  
group by Branch\_Code;

select sum(Total\_Fare) TotalFare from Reservation;

select Branch\_Code, sum(Total\_Fare) TotalFare, max(Total\_Fare) MaxFare,  
min(Total\_Fare) MinFare  
from Reservation  
group by Branch\_Code;

select Branch\_Code, sum(Total\_Fare) TotalFare, max(Total\_Fare) MaxFare,  
min(Total\_Fare) MinFare  
from Reservation  
where Flightno='AN56'  
group by Branch\_Code;

### **Having Clause Queries:-**

select Branch\_Code, sum(Total\_Fare) TotalFare, max(Total\_Fare) MaxFare,  
min(Total\_Fare) MinFare from Reservation  
where Flightno='AN56'  
group by Branch\_Code  
having Branch\_Code='SBI';

### **Group by Multiple Column Queries:-**

select Branch\_Code, Flightno, sum(Total\_Fare) TotalFare  
from Reservation  
group by Flightno,

Branch\_Code;

**Mathematical**

### **Function Queries:-**

use student;

select \* from Branch;  
select count(\*) TotalRows from Branch;

```
select * from Reservation;  
select count(*) TotalRows from Reservation;
```

```
select * from Airbus;  
select max(First_cap) FirstCap, min(Eco_cap) EcoCap from Airbus;
```

```
select * from Reservation;  
select avg(Total_Fare) Total_Fare from Reservation;  
select sum(Total_Fare) Total_Fare from Reservation;
```

```
select round(120.8333) Value;  
select round(120.5333) Value;  
select round(120.4333) Value;  
select truncate(120.8333,2) Value;  
select truncate(120.8333,0) Value;  
select truncate(120.8333,4) Value;  
select power(4,3) Value;  
select power(2,2) Value;  
select sqrt(4) Value;  
select * from Reservation;  
select Pnr, Pass_Name, IFNULL(Credit_Card_No,0) Value from Reservation; select  
IFNULL(CAST(Credit_Card_No AS Char), 'CASH') Value from Reservation;
```

### **Character Function Queries:-**

```
use student;
```

```
select * from Branch;  
select Branch_Code, length(Add1) Address1, length(Add2) Address2 from Branch;
```

```
select * from Fare;  
select Route_Code, substr(Route_Code,1,3) rtcd from Fare;
```

```
select replace('A001','A','B') ChangeData;  
select rtrim('A001 ') ChangeData;  
select ltrim(' A001') ChangeData;
```

```
select * from Branch;  
select Branch_Code, lower(Branch_Code) Branch_Code1 from Branch;  
select Add1, upper(Add1) Address from Branch;
```

```
select reverse('A001') ChangeData;  
select repeat("MYSQL ",5) ChangeData;  
select strcmp("MYSQL","MYSQL") ChangeData;  
select strcmp("SQL","MYSQL") ChangeData;  
select decode(encode('F','First'),'First') ChangeData;
```

**Conclusion:-** In this Practical I learn MYSQL command for Group By Clause, Statistical Built in Aggregate Functions and String Functions.

Godavari Foundation's  
Godavari College of Engineering, Jalgaon  
Department of Computer  
**Lab Manual**  
Database System Laboratory  
**Practical No:- 4**

**Date:-** 07-10-2020

**Name of Student :-** Shweta Ravindra Patil.

**Class:-** T.Y. Computer Engineering.

**Roll No:-** 34

**Title:-** Sub-queries, Set operations, Joins.

**Aim:-** MYSQL Commands for sub-queries, set Operations(UNION, UNION ALL, INTERSECT, MINUS) and Joins(Inner Join, Outer Join, Equi Join, Cross Join).

**Software Requirement :-** \_MYSQL.

**Hardware Requirement:-** 2GB RAM

**Theory:-**

**SUB QUERIES**

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.

**Subqueries with the SELECT Statement**

**Syntax:-**

```
SELECT column_name [, column_name ]  
FROM table1 [, table2 ]  
WHERE column_name OPERATOR  
(SELECT column_name [, column_name ]  
FROM table1 [, table2 ]  
[WHERE])
```



### **Example:-**

```
SQL> SELECT * FROM  
CUSTOMERS  
WHERE ID IN (SELECT ID  
FROM CUSTOMERS WHERE  
SALARY > 4500) ;
```

### **Subqueries with the INSERT Statement**

#### **Syntax:-**

```
INSERT INTO table_name [ (column1 [, column2 ]) ] SELECT [  
*|column1 [, column2 ]  
FROM table1 [, table2 ]  
[ WHERE VALUE OPERATOR ]
```

### **Example:-**

```
SQL>INSERTINTOCUSTOMERS_BKPSELECT  
*FROMCUSTOMERS WHEREIDIN  
(SELECTID  
FROM CUSTOMERS) ;
```

### **Subqueries with the UPDATE Statement**

#### **Syntax:-**

```
UPDATE table  
SET column_name = new_value  
[ WHERE OPERATOR [ VALUE ]  
(SELECT COLUMN_NAME FROM  
TABLE_NAME)  
[ WHERE) ]
```

### **Example:-**

```
SQL> UPDATE CUSTOMERS SET  
SALARY = SALARY * 0.25  
WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP WHERE  
AGE >= 27 );
```

### **Subqueries with the DELETE Statement**

#### **Syntax:-**

```
DELETE FROM TABLE_NAME [
WHERE OPERATOR[VALUE](SELEC
T COLUMN_NAME
FROM TABLE_NAME)
[ WHERE) ]
```

### **Example:-**

```
SQL> DELETE FROM CUSTOMERS
WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP WHERE
AGE >= 27 );
```

## **SET OPERATORS**

Set operators are used to join the results of two (or more) SELECT statements.

### **UNION**

When multiple SELECT queries are joined using UNION operator, it displays the combined result from all compounded SELECT queries, after removing all duplicates and in sorted order ( ascending by default), without allowing the NULL values.

### **Example:-**

```
SELECT 1 NUM
FROM DUAL UNION
SELECT 5
FROM DUAL UNION
SELECT 3
FROM DUAL UNION
SELECT 6
FROM DUAL UNION
SELECT 3 FROM DUAL;
```

### **Output:-**

```
NUM
1
3
5
6
```

### **UNION ALL**

UNION ALL gives the result set without removing duplication and sorting the data.

### **Example:-**

```
SELECT 1 NUM
```

```
FROM DUAL UNION ALL
SELECT 5
FROM DUAL UNION ALL
SELECT 3
FROM DUAL UNION ALL
SELECT 6
FROM DUAL UNION ALL
SELECT 3 FROM DUAL;
```

**Output:-**

```
NUM 1
5
3
6
3
```

**INTERSECT**

Using INTERSECT operator, Oracle displays the common rows from both the SELECT statements, with no duplicates and data arranged in sorted order (ascending by default).

**Example:-**

```
SELECT SALARY
FROM employees
WHERE DEPARTMENT_ID = 10
INTERSECT
SELECT SALARY
FROM employees WHERE DEPARTMENT_ID = 20;
```

**MINUS**

Minus operator displays the rows which are present in the first query but absent in the second query, with no duplicates and data arranged in ascending order by default.

**Example:-**

```
SELECT SALARY
FROM employees
WHERE DEPARTMENT_ID = 10 MINUS
SELECT SALARY
FROM employees
WHERE DEPARTMENT_ID = 20
```

## **JOINS**

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Different Types of SQL JOINS:-

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

### **Inner JOIN (Simple Join)**

It is used to return all rows from multiple tables where the join condition is satisfied. It is the most common type of join.

#### **Syntax:-**

```
SELECT columns  
FROM table1 INNER  
JOIN table2  
ON table1.column = table2.column;
```

#### **Example:-**

```
SELECT officers.officer_name, officers.address, students.course_name  
FROM officers  
INNER JOIN students  
ON officers.officer_id = students.student_id;
```

### **Left Outer Join**

The LEFT OUTER JOIN returns all rows from the left hand table specified in the ON condition and only those rows from the other table where the join condition is fulfilled.

#### **Syntax:-**

```
SELECT columns FROM  
table1  
LEFT[OUTER] JOIN table2  
ON table1.column = table2.column;
```

### **Example:-**

```
SELECT officers.officer_name, officers.address, students.course_name
FROM officers
LEFT JOIN students
ON officers.officer_id = students.student_id;
```

### **Right Outer Join**

The MySQL Right Outer Join returns all rows from the RIGHT-hand table specified in the ON condition and only those rows from the other table where the join condition is fulfilled.

### **Syntax:-**

```
SELECT columns FROM
table1
RIGHT [OUTER] JOIN table2
ON table1.column = table2.column;
```

### **Example:-**

```
SELECT officers.officer_name, officers.address, students.course_name, students.student_name FROM
officers
RIGHT JOIN students
ON officers.officer_id = students.student_id;
```

## **MYSQL QUERIES**

### **SUB QUERIES:-**

use student;

### **IN**

```
select Pass_Name from Reservation
where Flightno IN
(select Flightno from Flight_Sch where Flight_day1=1);
```

```
select Pnr, Flightno, FlightDate, Reserv_Date, Total_Fare, Branch_Code from
Reservation
where Branch_Code IN
(select Branch_Code from Branch where City='Jalgaon');
```

### **NOT IN**

```
select Pass_Name from Reservation
where Flightno NOT IN
(select Flightno from Flight_Sch where Flight_day1=1);
```

```
select Pnr, Flightno, FlightDate, Reserv_Date, Total_Fare, Branch_Code from
Reservation
where Branch_Code NOT IN
(select Branch_Code from Branch where City='Jalgaon');
```

### **EXISTS**

```
select * from Flight where
EXISTS
(select * from Reservation
where Reservation.Flightno=Flight.Flightno); select *
from Flight;
select Flightno from Reservation;
```

### **NOT EXISTS**

```
select * from Flight where
NOT EXISTS
( select * from Reservation
where Reservation.Flightno=Flight.Flightno);
```

### **ANY**

```
select Pass_Name, Total_Fare from
Reservation
where Total_Fare > ANY (select
Total_Fare
from Reservation
where Branch_Code='SBI');
```

### **ALL**

```
select Pass_Name, Total_Fare from
Reservation
where Total_Fare > ALL (select
Total_Fare
from Reservation
where Branch_Code='SBI');
```

### **SET OPERATIONS QUERIES:-**

```
use student;
```

### **UNION**

```
select Flightno, Flight_Date
from Flight
```

UNION

```
select Flightno, Flight_Date  
from Reservation
```

```
select * from Flight; select *  
from Reservation;
```

### **UNION ALL**

```
select Flightno, Flight_Date  
from Flight
```

UNION ALL

```
select Flightno, Flight_Date  
from Reservation
```

```
select * from Flight; select *  
from Reservation;
```

### **INTERSECT**

```
select Flightno, Flight_Date  
from Flight
```

INTERSECT

```
select Flightno, Flight_Date  
from Reservation;
```

```
select * from Flight; select *  
from Reservation;
```

### **ALIAS QUERY for INTERSECT in MYSQL:-**

```
select Flight.Flightno  
from Flight  
where Flight.Flightno IN  
(select Reservation.Flightno from Reservation);
```

### **MINUS**

```
select Flightno, Flight_Date  
from Flight
```

MINUS

```
select Flightno, Flight_Date  
from Reservation;
```

```
select * from Flight; select *  
from Reservation;
```

### **ALIAS QUERY for MINUS in MYSQL:-**

```
select Flightno
from Flight
LEFT JOIN Reservation
USING(Flightno)
where Reservation.Flightno IS NULL;
```

### **JOINS QUERIES:-**

use student;

### **INNER JOIN**

```
select Flight.Flightno, Flight_Date, Airbusno, Deprt_time
from Flight
INNER JOIN Flight_Sch
ON Flight.Flightno=Flight_Sch.Flightno;
```

```
select * from Flight;
select * from Flight_Sch;
```

### **LEFT OUTER JOIN**

```
select Flight.Flightno, Flight_Date, Airbusno, Deprt_time
from Flight
LEFT OUTER JOIN Flight_Sch
ON Flight.Flightno=Flight_Sch.Flightno;
```

```
select * from Flight;
select * from Flight_Sch;
```

### **LEFT JOIN**

```
select Flight.Flightno, Flight_Date, Airbusno, Deprt_time
from Flight
LEFT JOIN Flight_Sch
ON Flight.Flightno=Flight_Sch.Flightno;
```

```
select * from Flight;
select * from Flight_Sch;
```

### **RIGHT OUTERJOIN**

```
select Flight.Flightno, Flight_Date, Airbusno, Deprt_time
from Flight
RIGHT OUTER JOIN Flight_Sch
ON Flight.Flightno=Flight_Sch.Flightno;
```

```
select * from Flight;
```



```
select * from Flight_Sch;
```

### **RIGHT JOIN**

```
select Flight.Flightno, Flight_Date, Airbusno, Deprt_time  
from Flight  
RIGHT JOIN Flight_Sch  
ON Flight.Flightno=Flight_Sch.Flightno;
```

```
select * from Flight;  
select * from Flight_Sch;
```

### **FULL OUTER JOIN**

```
select Flight.Flightno, Flight_Date, Airbusno, Deprt_time  
from Flight  
FULL OUTER JOIN Flight_Sch  
ON Flight.Flightno=Flight_Sch.Flightno;
```

```
select * from Flight;  
select * from Flight_Sch;
```

### **FULL JOIN**

```
select Flight.Flightno, Flight_Date, Airbusno, Deprt_time  
from Flight  
FULL JOIN Flight_Sch  
ON Flight.Flightno=Flight_Sch.Flightno;
```

```
select * from Flight;  
select * from Flight_Sch;
```

### **EQUI JOIN**

```
select x.Route_Code, x.First_Fare, y.First_Fare from  
Fare x, Fare y  
where x.First_Fare=y.First_Fare;
```

```
select x.Route_Code, x.First_Fare, y.First_Fare from  
Fare x, Fare y  
where x.First_Fare < y.First_Fare and y.Route_Code='AUR-JAL';
```

### **CROSS JOIN**

```
select x.Flightno, x.Flight_Date, y.Route_Code from  
Flight x  
CROSS JOIN Flight_Sch y;
```

```
select * from Flight; select *  
from Flight_Sch;
```

**Conclusion:-** In this Practical I learn MYSQL Commands for sub-queries, set Operations(UNION, UNION ALL, INTERSECT, MINUS) and Joins(Inner Join, Outer Join, Equi Join, Cross Join).

Godavari Foundation's  
Godavari College of Engineering, Jalgaon  
Department of Computer  
**Lab Manual**  
Database System Laboratory  
**Practical No:- 5**

**Date:-** 19-10-2020

**Name of Student:-** Shweta Ravindra Patil.

**Class:-** T.Y. Computer Engineering.

**Roll No:-** 34

**Title:-** Creation of databases, writing SQL and PL/SQL queries to retrieve information from the databases.

**Aim:-** Database Connectivity using HTML, PHP, MYSQL, APACHE2

**Software Requirement :-** MYSQL.

**Hardware Requirement :-** 2GB RAM.

**Theory:-**

**Database Connectivity**

HTML, PHP, MYSQL, APACHE2

**Step 1: Filter HTML form requirements for form web page**

Step 1: Filter HTML form requirements for form web page Step

2: Create a database and a table in MYSQL

Step 3: Create HTML form

Step 4: Create HTML page to insert HTML form data in MYSQL database Step

5: All done! Select Field Name username, password, gender, email, phone Code, phone

**Step 2: Create a database and a table in MYSQL**

- 1) Type **mysql -u root -p** in command prompt. The prompt appear. As follow  
Enter password  
mysql> prompt appear.
- 2) Create database by name **Student**.
- 3) Open database by the command **use Student;** and create table register

```
mysql> use Student;
```

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> desc rigister;
```

```
+-----+-----+-----+-----+-----+
|Field      |Type      | Null | Key | Default | Extra|
+-----+-----+-----+-----+-----+
| username  | varchar(20) | YES  |     | NULL    |      |
| password  | varchar(20) | YES  |     | NULL    |      |
| gender    | varchar(10) | YES  |     | NULL    |      |
| email     | varchar(30) | YES  |     | NULL    |      |
| phoneCode | int(4)      | YES  |     | NULL    |      |
| phone     | bigint(12)  | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+
6 rows in set (0.02 sec)
```

### Step 3: Create HTML form

Create web page with name of “form.html” Type `sudo nano /var/www.html/form.html` . Copy and paste the following code.

```
<!DOCTYPE HTML>
<html>
<head>
    <title>Register Form</title>
</head>
<body bgcolor="pink" height="400px" width="400px">

    <form action="inset.php" method="POST">

<table>
    <tr>
        <td>Name :</td>
        <td><input type="text" name="username" required></td>
    </tr>
    <tr>
        <td>Password :</td>
        <td><input type="password" name="password" required></td>
    </tr>
    <tr>
        <td>Gender :</td>
        <td>
            <input type="radio" name="gender" value="m" required>Male
            <input type="radio" name="gender" value="f" required>Female
        </td>
    </tr>
    <tr>
        <td>Email :</td>
```

```

        <td><input type="email" name="email" required></td>
    </tr>
    <tr>
        <td>Phone no :</td>
        <td><select name="phoneCode" required>
            <option selected hidden value="">Select Code</option>
            <option value="977">977</option>
            <option value="978">978</option>
            <option value="979">979</option>
            <option value="973">973</option>
            <option value="972">972</option>
            <option value="974">974</option>
        </select>
        <input type="phone" name="phone" required>
    </td>
    </tr>
    <tr>
        <td><input type="submit" value="Submit"></td>
    </tr>
</table>
</form>
</body>
</html>

```

Test it in localhost link <http://localhost/form.html>

#### Step 4: Create PHP page to insert data entered in form.html webpage to MYSQL database

1) The “form.html” action is “inset.php” page. On this web page code is written for inserting record in database. For storing data in MYSQL database as records, first connect with database. For connect with database code is

```
$con=mysqli_connect("localhost","localhost_database_user","localhost_database_password",
"localhost_database_db");
```

2) Generally Local Host Database user is root and password is root or blank  
The code is as below

```
$con = mysqli_connect ('localhost','root','root','student');
```

3) “Student is database name that is create before. After connection to database , take post variable from the see the code below.

```

$username = $_POST['username'];
$password = $_POST['password'];
$gender = $_POST['gender'];
$email = $_POST['email'];
$phoneCode = $_POST['phoneCode'];
$phone = $_POST['phone'];

```

3) After Post variable write following SQL command.

```
$INSERT = "INSERT Into register (username, password, gender, email, phoneCode, phone)
values(?, ?, ?, ?, ?, ?)";
```

4) To execute Query on database, write the code

```
$stmt->execute();
```

5) inset.php file code below

```
<?php
$username = $_POST['username'];
$password = $_POST['password'];
$gender = $_POST['gender'];
$email = $_POST['email'];
$phoneCode = $_POST['phoneCode'];
$phone = $_POST['phone'];
if (!empty($username) || !empty($password) || !empty($gender) || !empty($email) ||
!empty($phoneCode))
    $host = "localhost";
    $dbUsername = "root";
    $dbPassword = "root";
    $dbname = "student";

    //create connection
    $conn = new mysqli($host, $dbUsername, $dbPassword, $dbname);

    if (mysqli_connect_error()) {
        die('Connect Error('. mysqli_connect_errno().').'. mysqli_connect_error());
    }
    else
    {
        $SELECT = "SELECT email From register Where email = ? Limit 1";
        $INSERT = "INSERT Into register (username, password, gender, email, phoneCode, phone)
values(?, ?, ?, ?, ?, ?$
        //Prepare statement
        $stmt = $conn->prepare($SELECT);
        $stmt->bind_param("s", $email);
        $stmt->execute();
        $stmt->bind_result($email);
        $stmt->store_result();
        $stmt->store_result();
        $stmt->fetch();
        $rnum = $stmt->num_rows;

        if ($rnum==0){
            $stmt->close();
            $stmt = $conn->prepare($INSERT);
            $stmt->bind_param("ssssii", $username, $password, $gender, $email, $phoneCode,
$phone);
```

```

        $stmt->execute();
        echo "New record inserted sucessfully";
    }
    else
    {
        echo "Someone already register using this email";
    }

    $stmt->close();
    $conn->close();
}
}
else
{
    echo "All field are required";
    die();
}
?>

```

### **Step 5: All done!**

- 1) Start **apache2** server.
- 2) Type Following UR in browser **<http://localhost/form.html>**
- 3) Fill the form details.
- 4) See record inserted in table in the database.

**Conclusion:-** In this Practical I learn Database Connectivity using HTML, PHP, MYSQL, APACHE2

Godavari Foundation's  
Godavari College of Engineering, Jalgaon  
Department of Computer  
**Lab Manual**  
Database System Laboratory  
**Practical No:- 6**

**Date:-** 29-10-2020

**Name of Student:-** Shweta Ravindra Patil.

**Class:-** T. Y. Computer Engineering.

**Roll No:-** 34

**Title:** Assignment on Triggers & Cursors.

**Aim:-** Program for Triggers and Cursors.

**Software Requirement:-** MYSQL.

**Hardware Requirement:-** 2GB RAM.

**Theory:-**

**TRIGGER**

A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

**Syntax:-**

```
create trigger [trigger_name]
[before | after]
{insert | update | delete} on
[table_name]
[for each row]
[trigger_body]
```



### **Explanation of syntax:-**

3.       create trigger [trigger\_name]: Creates or replaces an existing trigger with the trigger\_name.
4.       [before | after]: This specifies when the trigger will be executed.
5.       {insert | update | delete}: This specifies the DML operation.
6.       on [table\_name]: This specifies the name of the table associated with the trigger.
7.       [for each row]: This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected.
8.       [trigger\_body]: This provides the operation to be performed as trigger is fired

### **BEFORE and AFTER Trigger:-**

BEFORE triggers run the trigger action before the triggering statement is run.

AFTER triggers run the trigger action after the triggering statement is run.

### **CURSOR**

**Cursor** is a Temporary Memory or Temporary Work Station. It is Allocated by Database Server at the Time of Performing DML operations on Table by User. Cursors are used to store Database Tables. There are 2 types of Cursors: Implicit Cursors, and Explicit Cursors. These are explained as following below.

#### **Implicit Cursors:-**

Implicit Cursors are also known as Default Cursors of SQL SERVER. These Cursors are allocated by SQL SERVER when the user performs DML operations.

#### **Explicit Cursors:-**

Explicit Cursors are Created by Users whenever the user requires them. Explicit Cursors are used for Fetching data from Table in Row-By-Row Manner.

### **How to create Explicit Cursor:-**

- **Declare Cursor Object.**

#### **Syntax :-**

```
DECLARE cursor_name CURSOR FOR SELECT * FROM table_name
```

```
DECLARE s1 CURSOR FOR SELECT * FROM studDetails
```

- **Open Cursor Connection.**

#### **Syntax:-**

```
OPEN cursor_connection
```

```
OPEN s1
```

- **Fetch Data from cursor.**

There are total 6 methods to access data from cursor.

They are as follows :-

**FIRST** is used to fetch only the first row from cursor table.

**LAST** is used to fetch only last row from cursor table.

**NEXT** is used to fetch data in forward direction from cursor table.

**PRIOR** is used to fetch data in backward direction from cursor table.

**ABSOLUTE n** is used to fetch the exact nth row from cursor table.

**RELATIVE n** is used to fetch the data in incremental way as well as decremental way.

**Syntax :-**

FETCH NEXT/FIRST/LAST/PRIOR/ABSOLUTE n/RELATIVE n FROM cursor\_name FETCH

FIRST FROM s1

FETCH LAST FROM s1

FETCH NEXT FROM s1 FETCH

PRIOR FROM s1 FETCH

ABSOLUTE 7

FROM s1 FETCH RELATIVE -2

FROM s1

**14. Close cursor connection.**

**Syntax :-**

CLOSE cursor\_name

CLOSE s1;

**15. Deallocate cursor memory.**

**Syntax :-**

DEALLOCATE cursor\_name

DEALLOCATE s1

**SOURCE CODE TRIGGER**

Create Trigger Flight\_Update

After Insert on Reservation For

Each Row

Begin

    If new.Class='F'

    Then Update Flight

    Set First\_Seats\_Bk = First\_Seats\_Bk + 1

    Where Flightno = new.Flightno and Flight\_Date = new.Flight\_Date;

    ELSEIF new.Class='B' Then

```

Update Flight
Set Bus_Seats_Bk=Bus_Seats_Bk + 1
Where Flightno = new.Flightno and Flight_Date = new.Flight_Date;
ELSEIF new.Class='E' Then
Update Flight
Set Eco_Seats_Bk = Eco_Seats_Bk + 1
Where Flightno = new.Flightno and Flight_Date = new.Flight_Date;
END IF;
END
//

```

## **SOURCE CODE CURSOR**

```

create procedure curproc()
begin
DECLARE mpnr int(5); DECLARE
mflightno CHAR(4); DECLARE
mflight_date date; DECLARE
mreserv_date date;
DECLARE mpass_name VARCHAR(20); DECLARE
mclass char(1);

DECLARE cursor_Reserv CURSOR
FOR SELECT Pnr,Flightno,Flight_Date,Reserv_Date,Pass_Name,Class FROM
Reservation where Class='F';

OPEN cursor_Reserv;

FETCH NEXT FROM cursor_Reserv INTO mpnr,
mflightno,
mflight_date,
mreserv_date,
mpass_name,
mclass;

LOOP
insert into Passengers values(mpnr,mflightno,mflight_date,mreserv_date,mpass_name,mclass);
FETCH NEXT FROM cursor_Reserv INTO mpnr,
mflightno,
mflight_date,
mreserv_date,
mpass_name,
mclass;

End LOOP;
close cursor_Reserv;
end
//

```

**Conclusion:-** In this Practical I learn Program for Triggers and Cursors.

Godavari Foundation's  
Godavari College of Engineering, Jalgaon  
Department of Computer  
**Lab Manual**  
Database System Laboratory  
**Practical No:- 7**

**Date:-** 07-11-2020

**Name of Student:-** Shweta Ravindra Patil.

**Class:-** T. Y. Computer Engineering.

**Roll No:-** 34

**Title:-** Normal Forms: First, Second, Third and Boyce Codd Normal Forms.

**Aim:-** Designing Normal Forms: First, Second, Third and Boyce Codd Normal Forms.

**Software Requirement :-** No Requirements.

**Hardware Requirement :-** No Requirements.

**Theory:-**

**Normalization**

Normalization in DBMS is the process of effectively organizing data into multiple relational tables to minimize data redundancy.

A functional dependency defines the relationship between two attributes, typically between a prime attribute (primary key) and non-prime attributes. If for a table X containing attributes A and B, among which the attribute A is a primary key then the functional dependency is defined as:

**Example for Functional Dependency:-**



Employee Table					
EMP_ID	EMP_NAME	EMP_CONTACT	MANAGER	Manager_Contact	DEPARTMENT
1	Oliver	88998899	Mr.X	33221133	Accounts
2	Barry	66776677	Mr.X	33221133	Accounts
3	Clark	44554455	Mr.Y	99887799	Sales
4	Bruce	22332233	Mr.X	33221133	Accounts
5	Kara	11001100	Mr.Y	99887799	Sales

### 9. Insertion Anomaly

Caused by updating the same set of repeated information again and again. This becomes a problem as the entries for a table increases with time.

### 10. Deletion Anomaly

It causes loss of data within the database due to its removal in some other related data set.

### 11. Updating Anomaly

In case of an update, it's very crucial to make sure that the given update happens for all the rows associated with the change. Even if a single row gets missed out it will lead to inconsistency of data.

For the normalization process to happen it is important to make sure that the data type of each data throughout an attribute is the same and there is no mix up within the data types.

## Types of Normal Forms

### 1<sup>st</sup> Normal Form (1NF)

Data Atomicity is maintained. The data present in each attribute of a table cannot contain more than one value.

For each set of related data, a table is created and data set value in each is identified by a primary key applicable to the data set.

### Example:-

Employee Table (Before 1NF)			Employee Table (After 1NF)		
EMP_ID	EMP_NAME	EMP_CONTACT	EMP_ID	EMP_NAME	EMP_CONTACT
1	Oliver	88998899, 34534554	1	Oliver	88998899
2	Barry	66776677	1	Oliver	34534554
3	Clark	44554455	2	Barry	66776677
4	Bruce	22332233, 98798797	3	Clark	44554455
5	Kara	11001100	4	Bruce	98798797
			4	Bruce	98798797
			5	Kara	11001100

The multiple values from EMP\_CONTACT made atomic based on the EMP\_ID to satisfy the 1NF rules.

## 2<sup>nd</sup> Normal Form (2NF)

The table should be in its 1NF,

The table should not have any partial dependencies

Employee Table (Before 2NF)		
EMP_ID	PROJECT_ID	MANAGER
1	23	Mr.X
1	67	Mr.Z
2	45	Mr.X
3	78	Mr.Y
3	23	Mr.X
4	23	Mr.X
5	78	Mr.Y
5	67	Mr.Z

If manager details are to be fetched for an employee, multiple results are returned when searched with EMP\_ID, in order to fetch one result, EMP\_ID and PROJECT\_ID together are considered as the Candidate Keys.

Here the manager depends on PROJECT\_ID and not on EMP\_ID, this creates a partial dependency.

There are multiple ways to eliminate this partial dependency and reduce the table to its 2<sup>nd</sup> normal form.

Employee Table		Project Table		
EMP_ID	PROJECT_ID	PROJECT_ID	PROJECT	MANAGER
1	23	23	Extract	Mr.X
1	67	67	Load	Mr.Z
2	45	45	Extract	Mr.X
3	78	78	Transform	Mr.Y
3	23	23	Extract	Mr.X
4	23	23	Extract	Mr.X
5	78	78	Transform	Mr.Y
5	67	67	Load	Mr.Z

### 3<sup>rd</sup> Normal Form (3NF)

- The table should be in its 2NF, and
- The table should not have any transitive dependencies

Employee Table (Before 3NF)			
EMP_ID	PROJECT_ID	PERFORMANCE_SCORE	HIKE (\$)
1	23	50	160
1	67	30	120
2	45	50	160
3	78	60	170
3	23	30	120
4	23	20	110
5	78	60	170
5	67	20	110

- The PERFORMANCE\_SCORE depends on both employee and project that he is associated, but the last column HIKE depends on PERFORMANCE\_SCORE.
- The hike changes with performance and here the attribute performance score is not a primary key. This forms a transitive dependency
- To eliminate the transitive dependency created, and to satisfy the 3NF, the table is broken as:



Employee Table			Performance Table	
EMP_ID	PROJECT_ID	PERFORMANCE_SCORE	PERFORMANCE_SCORE	HIKE (\$)
1	23	50	20	110
1	67	30	30	120
2	45	50	50	160
3	78	60	60	170
3	23	30		
4	23	20		
5	78	60		
5	67	20		

### Boyce-Codd Normal Form BCNF or 3.5 NF

16. The table should be in its 3 NF form
17. For any dependency,  $A \rightarrow B$ , A should be a super key i.e. A cannot be an on prime attribute when B is a prime attribute.

Employee Table (Before BCNF)		
EMP_ID	PROJECT_ID	DEPARTMENT
1	23	Accounts
1	67	Sales
2	45	HR
3	78	Accounts
3	23	Sales
4	23	HR
5	78	Sales
5	67	HR

1.  $EMP\_ID + PROJECT\_ID \rightarrow DEPARTMENT$  (Candidate keys)
2. DEPARTMENT which is not a super key is dependent only on PROJECT\_ID but not dependent on EMP\_ID
3.  $Department \rightarrow Project\_ID$
4. (Non-prime attribute) (prime attribute)

To make this table satisfy BCNF, we need to break the table as:

After BCNF

Employee Table

EMP_ID	DEPARTMENT_ID
1	D123
1	D456
2	D789
3	D122
3	D455
4	D789
5	D454
5	D787

Department Table

DEPARTMENT_ID	DEPARTMENT	PROJECT	PROJECT_ID
D123	Accounts	Extract	23
D456	Sales	Load	67
D789	HR	Extract	45
D122	Accounts	Transform	78
D455	Sales	Extract	23
D789	HR	Extract	23
D454	Sales	Transform	78
D787	HR	Load	67

**Conclusion:-** In this Practical I learn Normal Forms: First, Second, Third and Boyce Codd Normal Forms.

Godavari Foundation's  
Godavari College of Engineering, Jalgaon  
Department of Computer  
**Lab Manual**  
Database System Laboratory  
**Practical No:- 8**

**Date:-** 23-11-2020

**Name of Student:-** Shweta Ravindra Patil.

**Class:-** T. Y. Computer Engineering.

**Roll No:-** 34

**Title:-** Assignment in Design and Implementation of Database systems or packages for applications such as office automation, hotel management, hospital management.

**Aim:-** Design and Implementation of Database systems or packages for hospital management.

**Software Requirement :-** MYSQL.

**Hardware Requirement :-** 2GB RAM.

**Theory:-**

**Hospital Management System**

A database is a collection of information and is systematically stored in the form of row and columns. The table in the database has unique name that identifies its contents. The database in turn is further described in detail giving all the fields used with the data types, constraints available, primary key and foreign key.

Database design is used to manage large bodies of information.

**Data types and its description:-**

Fields in database table have a data type. Some of the data types used in database table are explained below.

**Integer:-**

One optional sign character (+ or -) followed by atleast one digit (0-9). Leading and trailing blanks are ignored. No other character is allowed.

### **Varchar:-**

It is used to store alpha numeric characters. In this data type we can set the maximum number of characters up to 8000 ranges by default SQL server will set the size to 50 characters large.

### **Date/Time:-**

Date/Time data type is used for representing data or time.

### **Patient Table:-**

Fields	Data Type	Relationships
Pid	Varchar(5)	Primary Key
Name	Varchar(20)	Not Null
Age	int	Not Null
Weight	int	Not Null
Gender	Varchar(10)	Not Null
Address	Varchar(50)	Not Null
Phoneno	int	Not Null
Disease	Varchar(20)	Not Null
Doctored	Varchar(5)	Not Null

### **Doctor Table:**

Fields	Data Type	Relationship
doctorid	Varchar(5)	Primary Key
doctorname	Varchar(15)	Not Null
dept	Varchar(15)	Not Null

### **LAB Table :**

Fields	Data Type	Relationships
labno	Varchar(5)	Primary Key
pid	Varchar(5)	Not Null
weight	int	Not Null
doctorid	Varchar(5)	Foreign Key
date	Date/Time	Not Null
category	Varchar(15)	Not Null
patient_type	Varchar(15)	Not Null
amount	int	Not Null

### **Inpatient Table:-**

Fields	Data Type	Relationships
pid	Varchar(5)	Primary Key
room_no	Varchar(50)	Not Null
date_of_adm	Date/Time	Not Null
date_of_dis	Date/Time	Not Null
advance	int	Not Null
labno	Varchar(5)	Foreign Key

**Outpatient Table:-**

Fields	Data Type	Relationships
pid	Varchar(5)	Primary Key
date	Date/Time	Not Null
labno	Varchar(5)	Foreign Key

**Room Table:-**

Fields	Data Type	Relationships
room_no	Varchar(50)	Primary Key
room_type	Varchar(10)	Not Null
status	Varchar(10)	Not Null

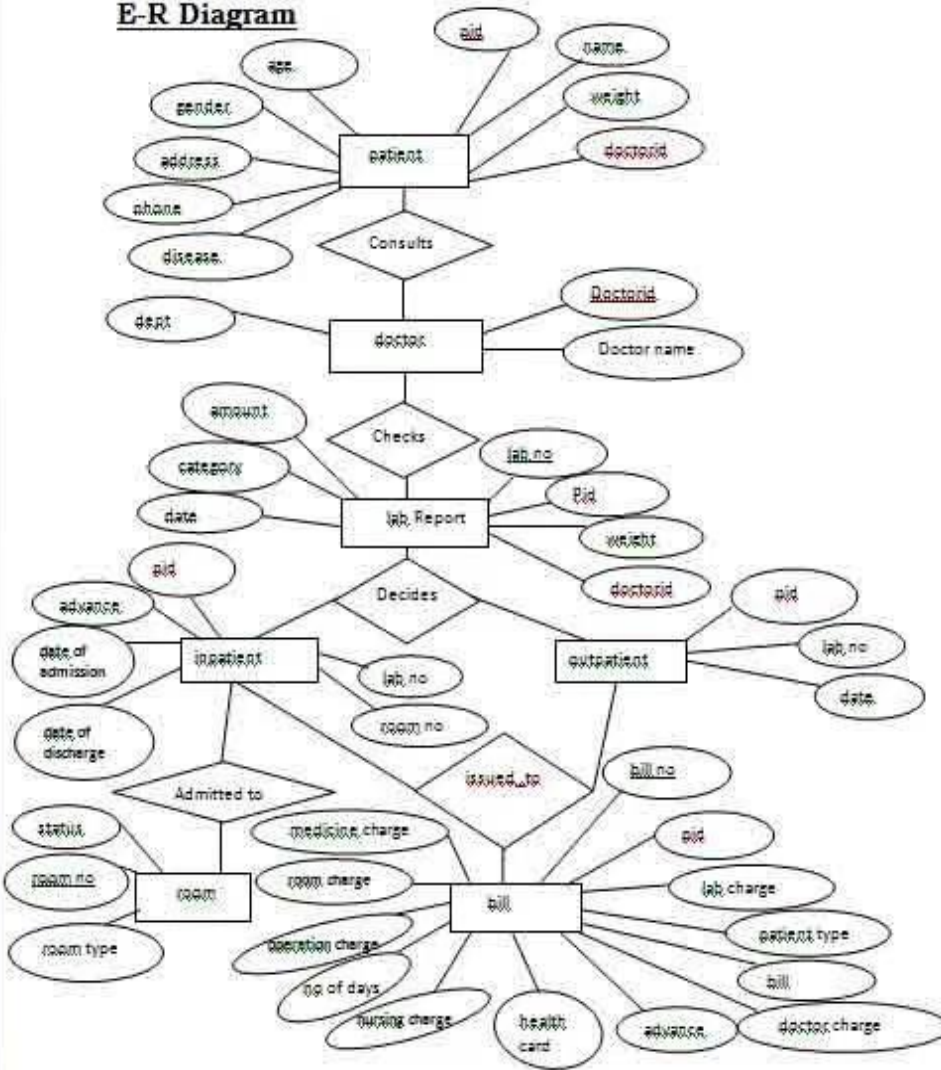
**Bill Table:-**

Fields	Data Type	Relationships
bill_no	Varchar(50)	Primary Key
pid	Varchar(5)	Foreign Key
patient_type	Varchar(10)	Allow Null
doctor_charge	int	Not Null
medicine_charge	int	Not Null
room_charge	int	Not Null
oprtn_charge	int	Allow Null
no_of_days	int	Allow Null
nursing_charge	int	Allow Null
advance	int	Allow Null
health_card	Varchar(50)	Allow Null
lab_charge	int	Allow Null
bill	int	Not Null

**E-R Diagram:-**

Entity relationship diagram is used in modern database software engineering to illustrate logical structure of database. It is a relational schema database modeling method used to model a system and approach. This approach commonly used in database design. The diagram created using this method is called E-R diagram.

## E-R Diagram



## MYSOL QUERIES

mysql> create table Patient

- > (Pid Varchar(5) Primary Key,
- > Name Varchar(20) Not Null,
- > Age int Not Null,
- > Weight int Not Null,
- > Gender Varchar(10) Not Null,
- > Adress Varchar(50) Not Null,
- > Phoneno int Not Null,
- > Disease Varchar(20) Not Null,
- > Doctored Varchar(5) Not Null);

mysql> create table Doctor(

- > doctorid Varchar(5) Primary Key,
- > doctorname Varchar(15) Not Null,
- > dept Varchar(15) Not Null);

```
mysql> create table Lab( labno Varchar(5) Primary Key,  
->pid Varchar(5) Not Null,  
->weight int NotNull,  
->doctoridVarchar(5),  
->date Date Not Null,  
->category Varchar(15) Not Null,  
->patient_type Varchar(15) Not Null,  
->amount int Not Null,  
->constraint doctorid_fk Foreign Key(doctorid) references Doctor(doctorid));
```

```
mysql> create table Inpatient(  
->pid Varchar(5) Primary Key,  
->room_no Varchar(50) Not Null,  
->date_of_adm Date Not Null,  
->date_of_dis Date Not Null,  
->advance int Not Null,  
->labno Varchar(5),  
->constraint labno_fk Foreign Key(labno) references Lab(labno));
```

```
mysql> create table Outpatient(  
->pid varchar(5) Primary Key,  
->date Date Not Null,  
->labno Varchar(5),  
->constraint labno_fk1 Foreign Key(labno) references Lab(labno));
```

```
mysql> create table Room(  
-> room_no Varchar(50) Primary Key,  
-> room_type Varchar(10) Not Null,  
-> status Varchar(10) Not Null);
```

```
mysql> create table Bill(  
-> bill_no Varchar(50) Primary Key,  
-> pid Varchar(5),  
-> patient_type Varchar(10),  
-> doctor_charge int Not Null,  
-> medicine_charge int Not Null,  
-> room_charge int Not Null,  
-> oprtn_charge int,  
-> no_of_days int,  
-> nursing_charge int,  
-> advance int,  
-> health_card Varchar(50),  
-> lab_charge int,  
-> bill int Not Null,  
-> constraint pid_fk Foreign Key(pid) references Inpatient(pid));
```

**Conclusion:-** In this Practical I learn Design and Implementation of Database systems or packages for hospital management.

Godavari Foundation's  
**Godavari College of Engineering, Jalgaon**  
Department of Computer  
**Lab Manual**  
Database System Laboratory  
**Practical No:- 9**

**Date:-** 27-11-2020

**Name of Student:-** Shweta Ravindra Patil.

**Class:-** T. Y. Computer Engineering.

**Roll No:-** 34

**Title:-** Deployment of Forms, Reports Normalization, Query Processing Algorithms in the above application project.

**Aim:-** Deploying web page on Red Hat JBoss EAP Server 7.0

**Software Requirement:-** Redhat Developer Studio , Redhat JBOSS EAP Server 7.0.

**Hardware Requirement:-** 2GB RAM.

**Theory:-**

**Deploying form on JBOSS EAP Server**

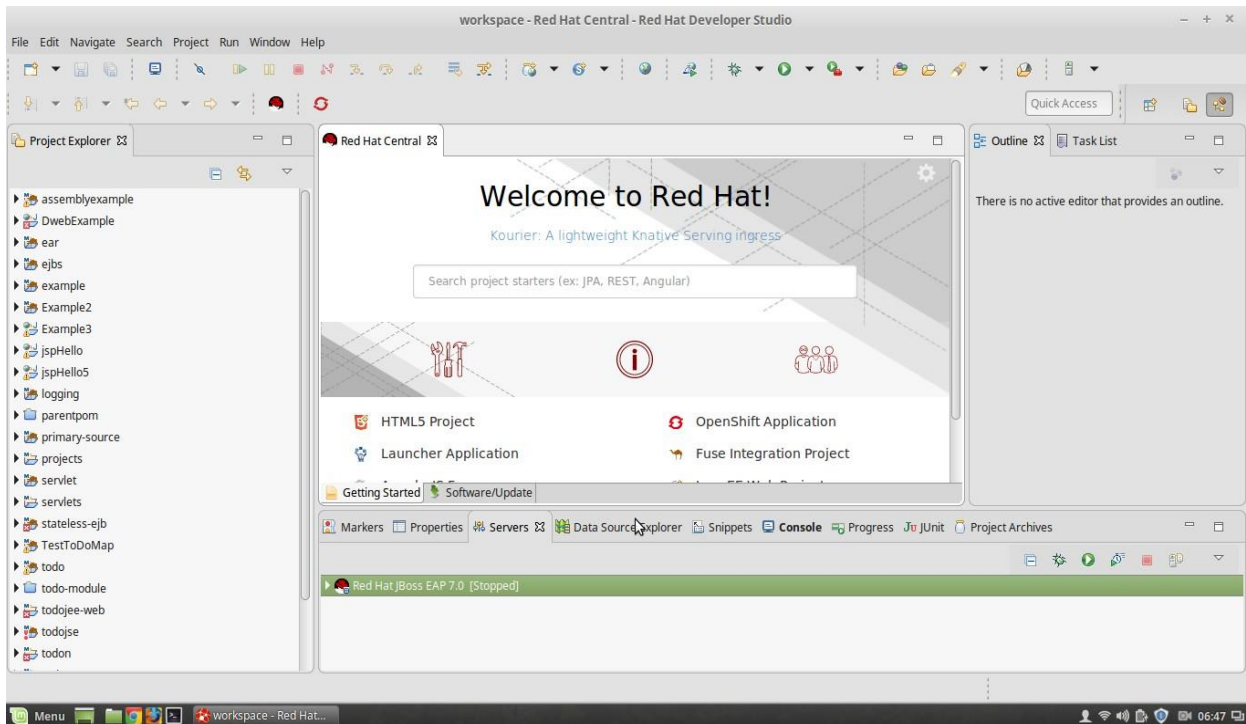
**Three type of Files are created:-**

- 18. JSP
- 19. XML
- 20. JAR



## **Step by Step Execution as shown below:-**

### **1) Start Red Hat JBOSS EAP 7.0 Server**



Click File -> New-> Dynamic Web Project

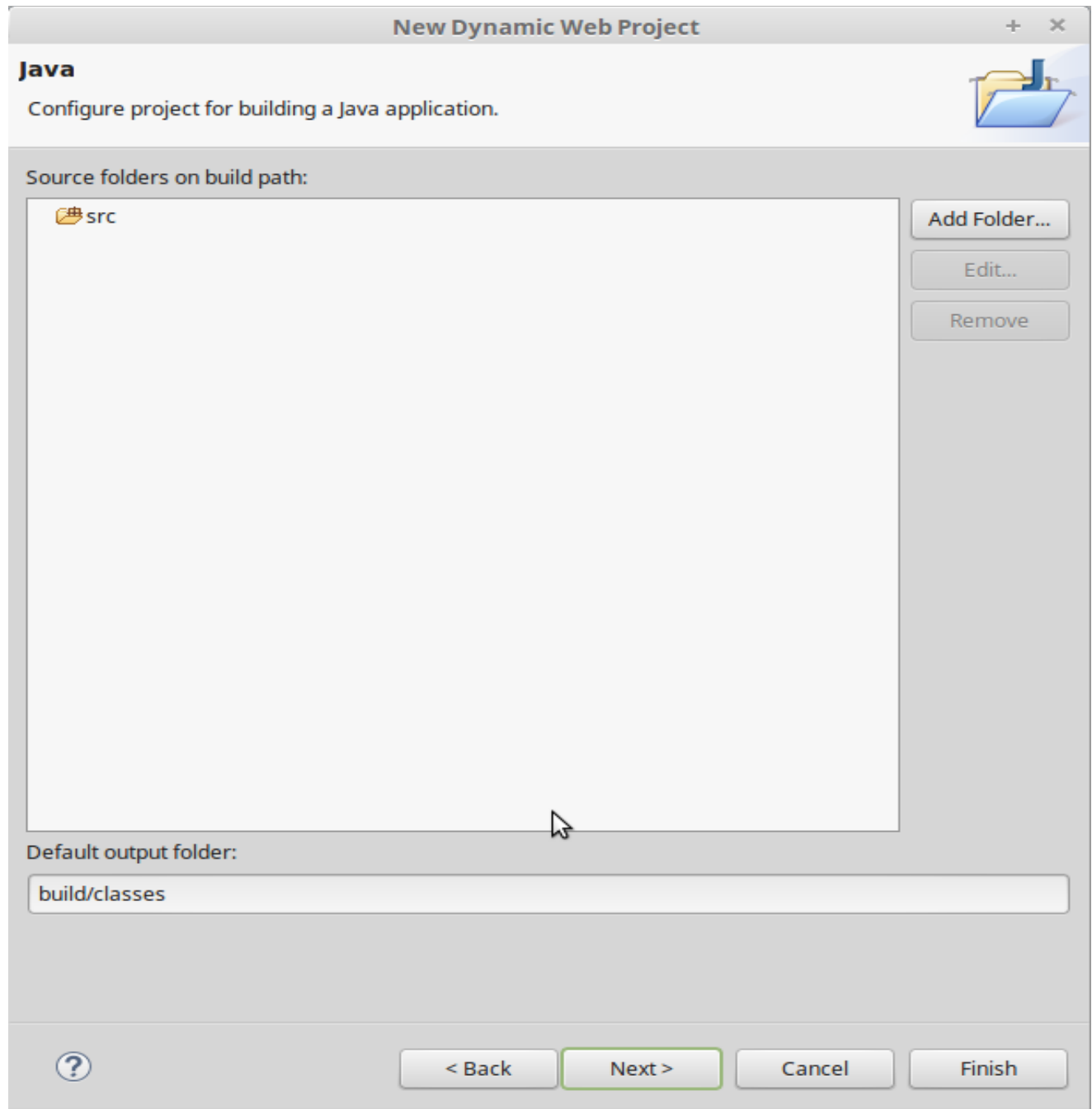
Enter Project Name

Select Minimal Configuration from Configuration

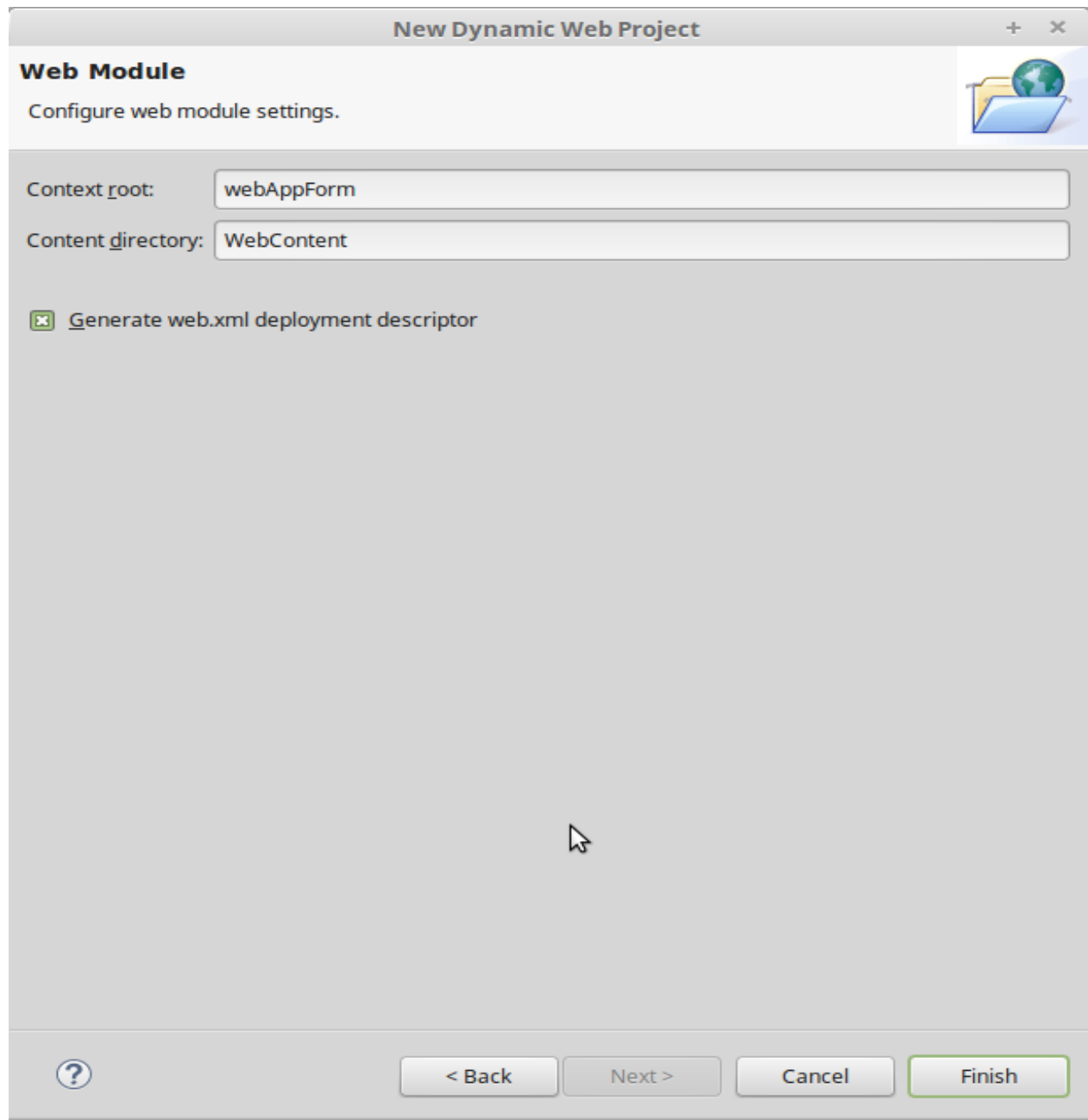
Select Add project to working sets

Click Next >

2) Select **src** Click Next >



3) Select Generate web.xml Deployment descriptor. Click Finish.




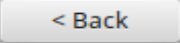
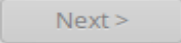
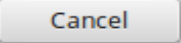

**New Dynamic Web Project**

**Web Module**  
Configure web module settings.

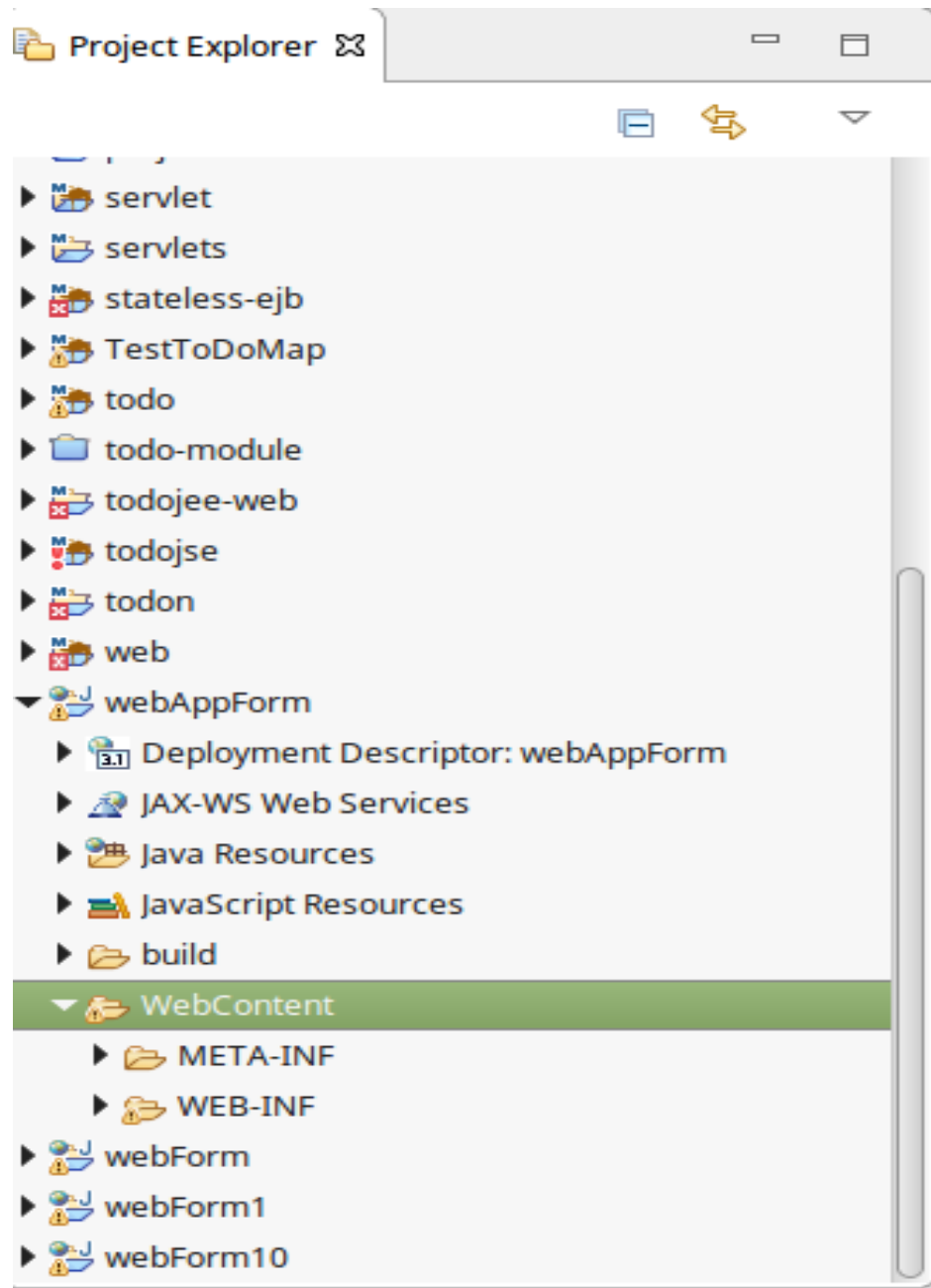
Context root: webAppForm

Content directory: WebContent

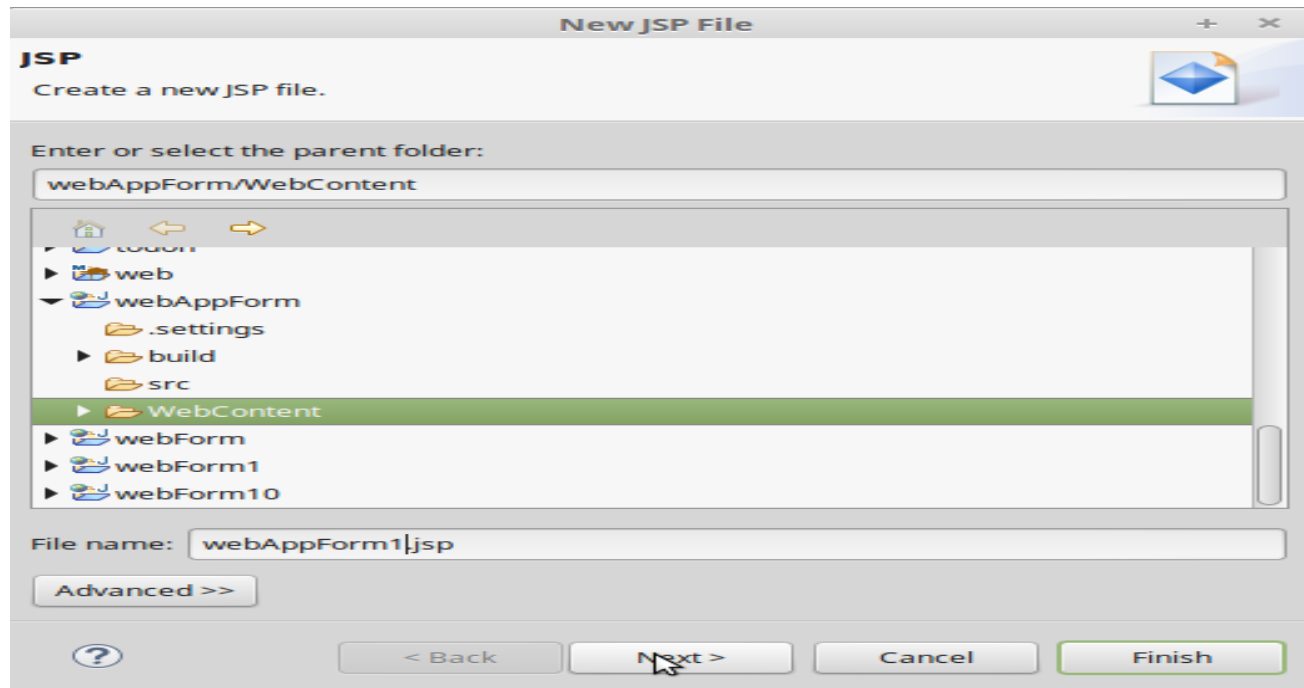
☒ Generate web.xml deployment descriptor

4) In Project Explorer webAppForm is create Right Click WebContent → New → JSP File.



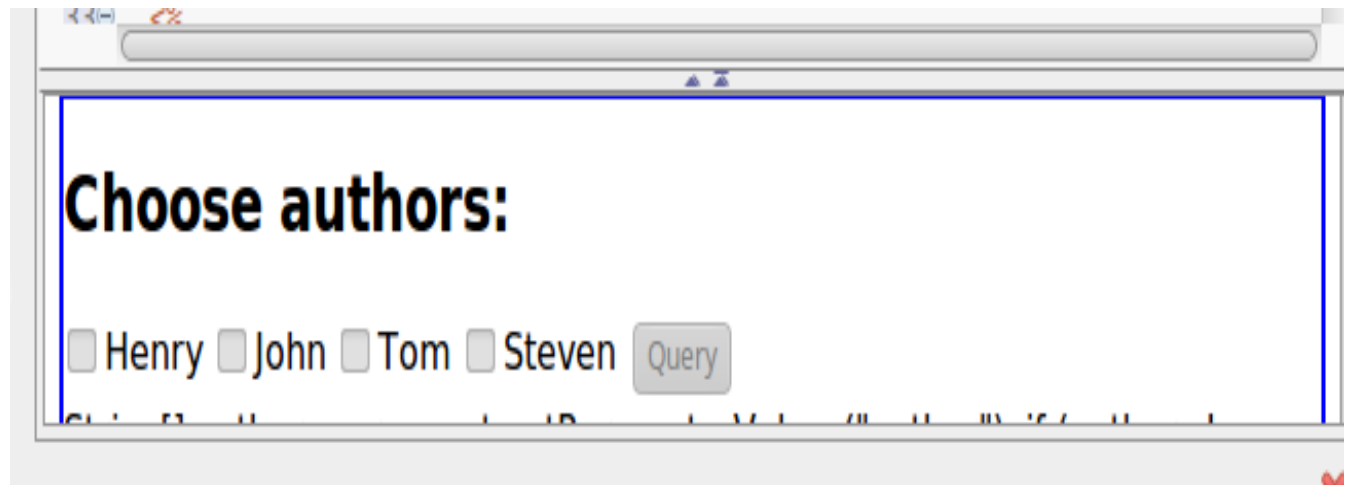
5 ) Enter jsp File Name. Click Finish. Open this jsp File and Write code. Open WebAppForm.jsp.



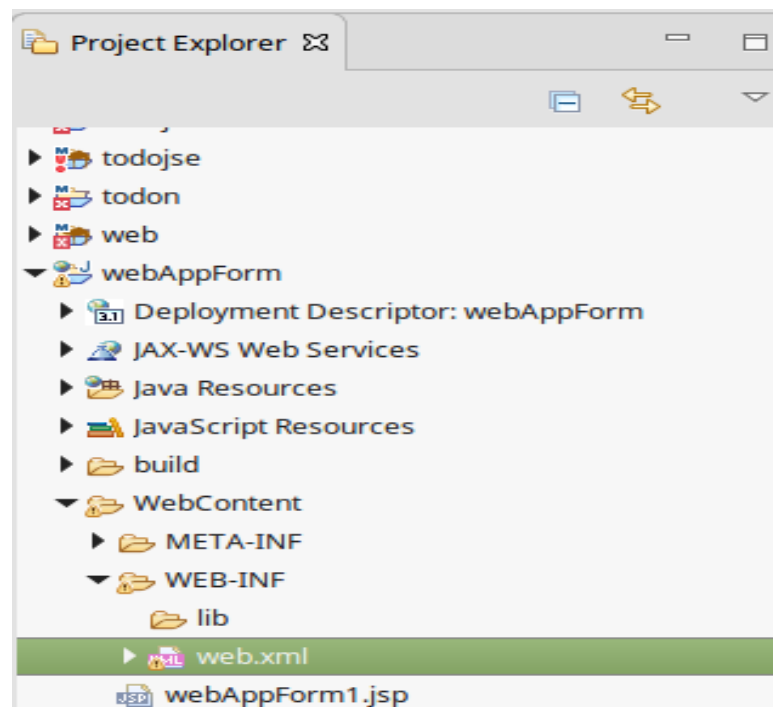
```
<h2>Choose authors:</h2>
<form method="get">
  <input type="checkbox" name="author" value="Henry Korth">Henry
  <input type="checkbox" name="author" value="John C. Martin">John
  <input type="checkbox" name="author" value="Tom Mitchell">Tom
  <input type="checkbox" name="author" value="Steven Halim">Steven
  <input type="submit" value="Query">
</form>

<%
String[] authors = request.getParameterValues("author");
if (authors != null) {
%>
  <h3>You have selected author(s):</h3>
  <ul>
    <%
      for (String author : authors) {
    %>
      <li><%= author %></li>
    <%
      }
    %>
  </ul>
<%
}
%>
<br /><a href="<%= request.getRequestURI() %>">BACK</a> |
```

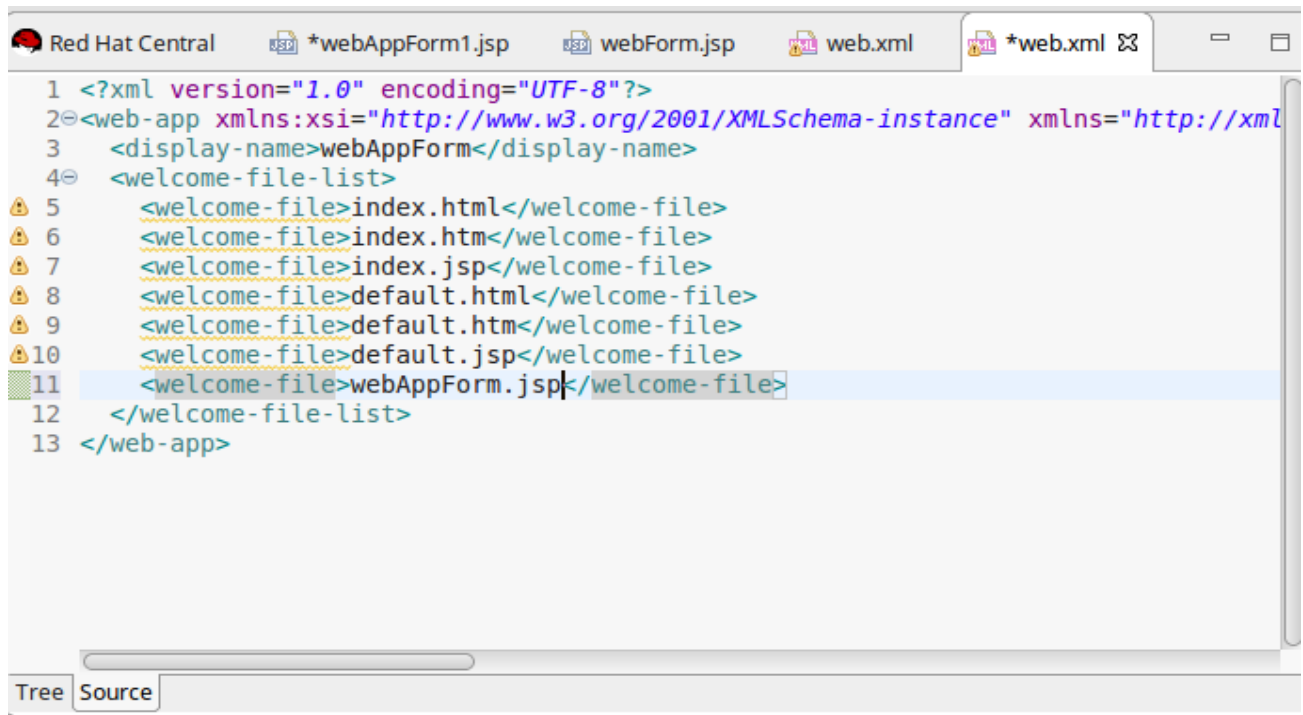
## 6) Output Screen:



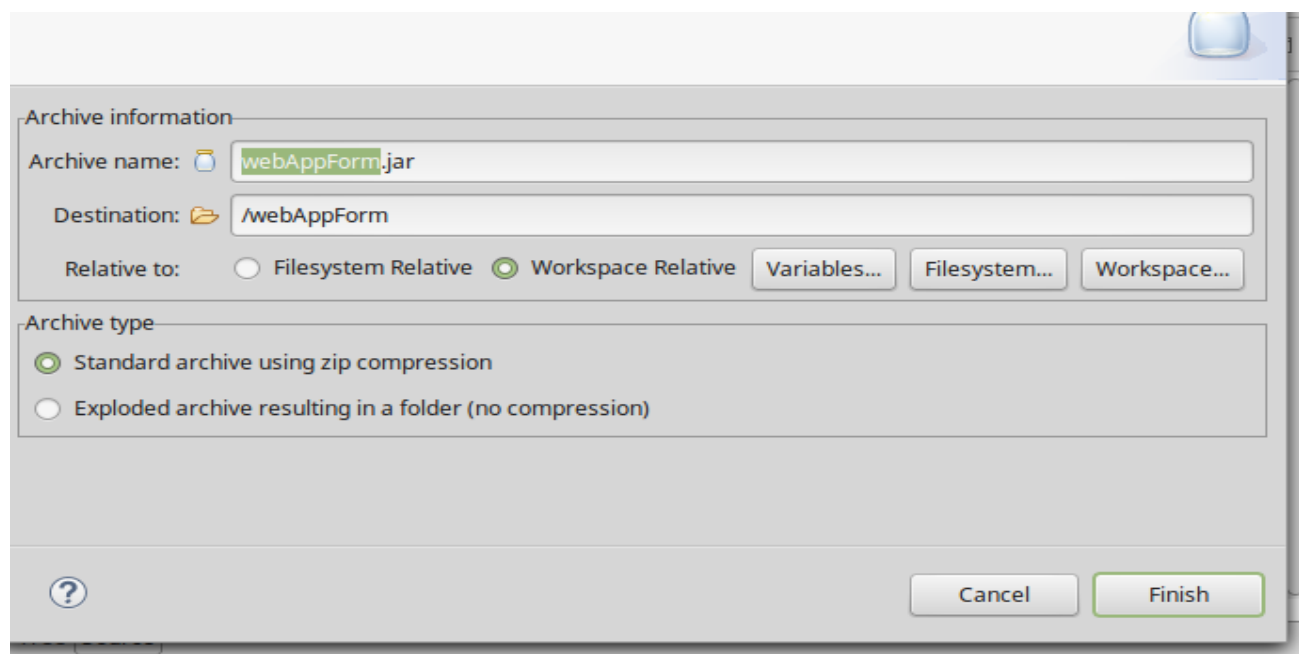
7 ) Open web.xml. Write name of jsp file enclose within <welcome-file><welcome-file>. Single Line is added to the code. <welcome-file> webAppForm.jsp </welcome-file>



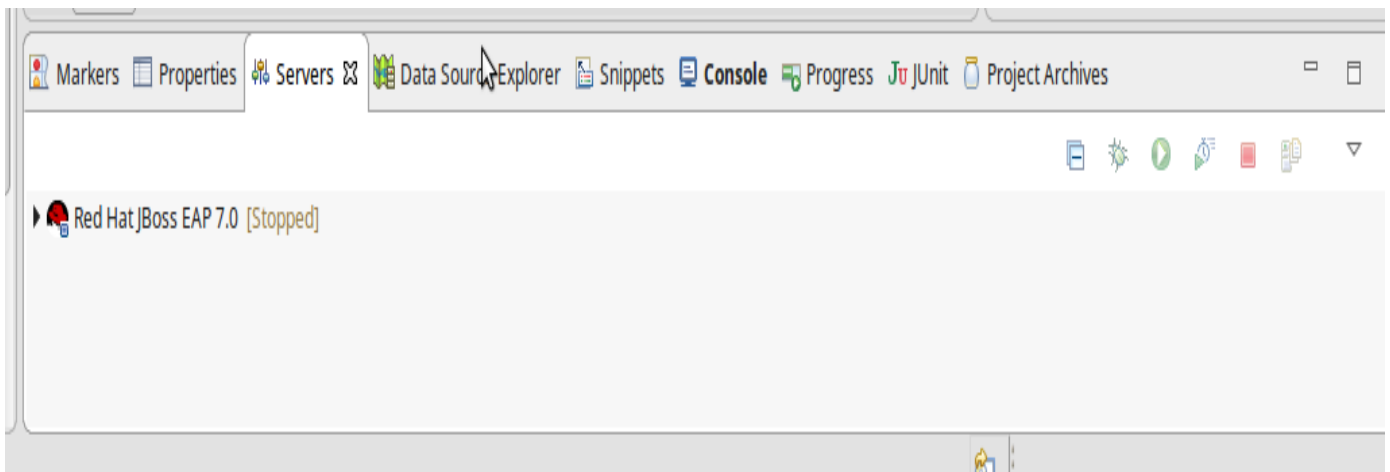
8) Goto Menubar. Select Windows → Show View → Other → Project Archives.



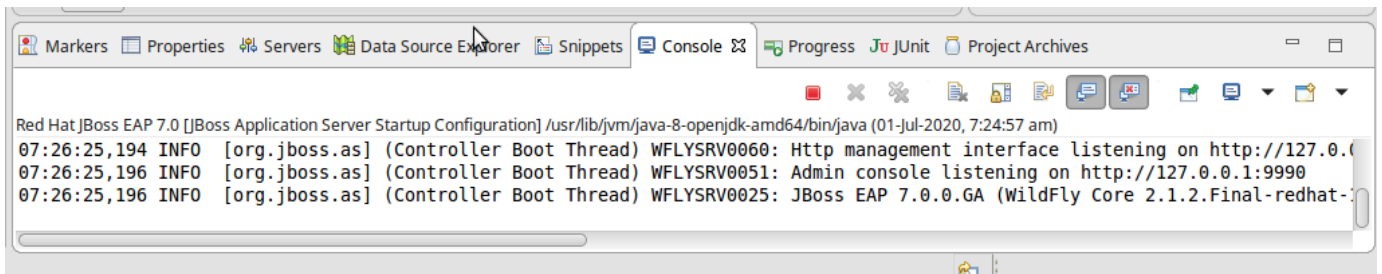
9) Click Open. Click Project Archives Tab. Right Click New Archive→ JAR.



10) Click Finish. Click Servers Tab. Red Hat JBOSS EAP 7.0 [ Stopped ]

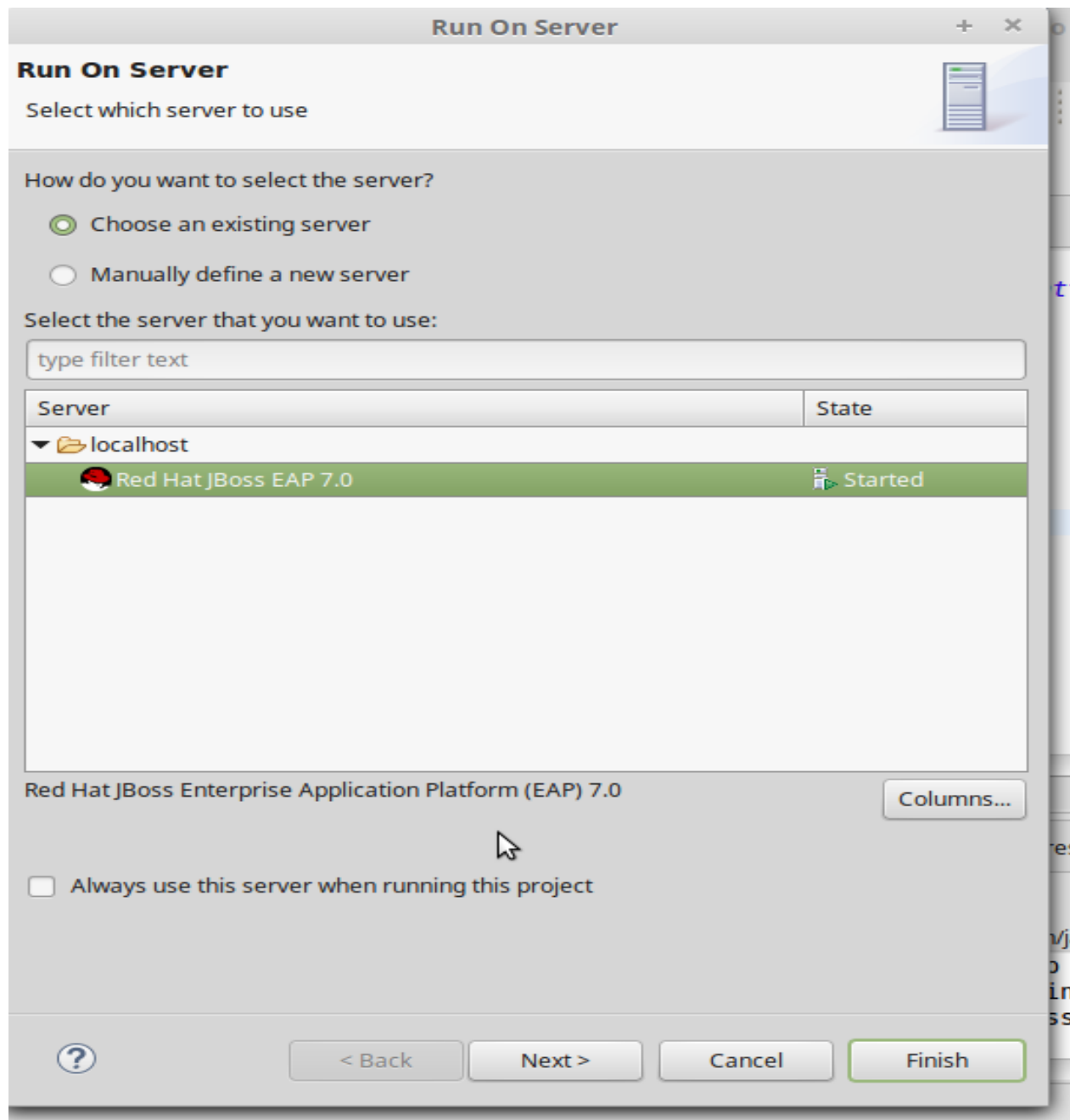


11) Right Click → Red Hat JBOSS EAP 7.0 → Start.

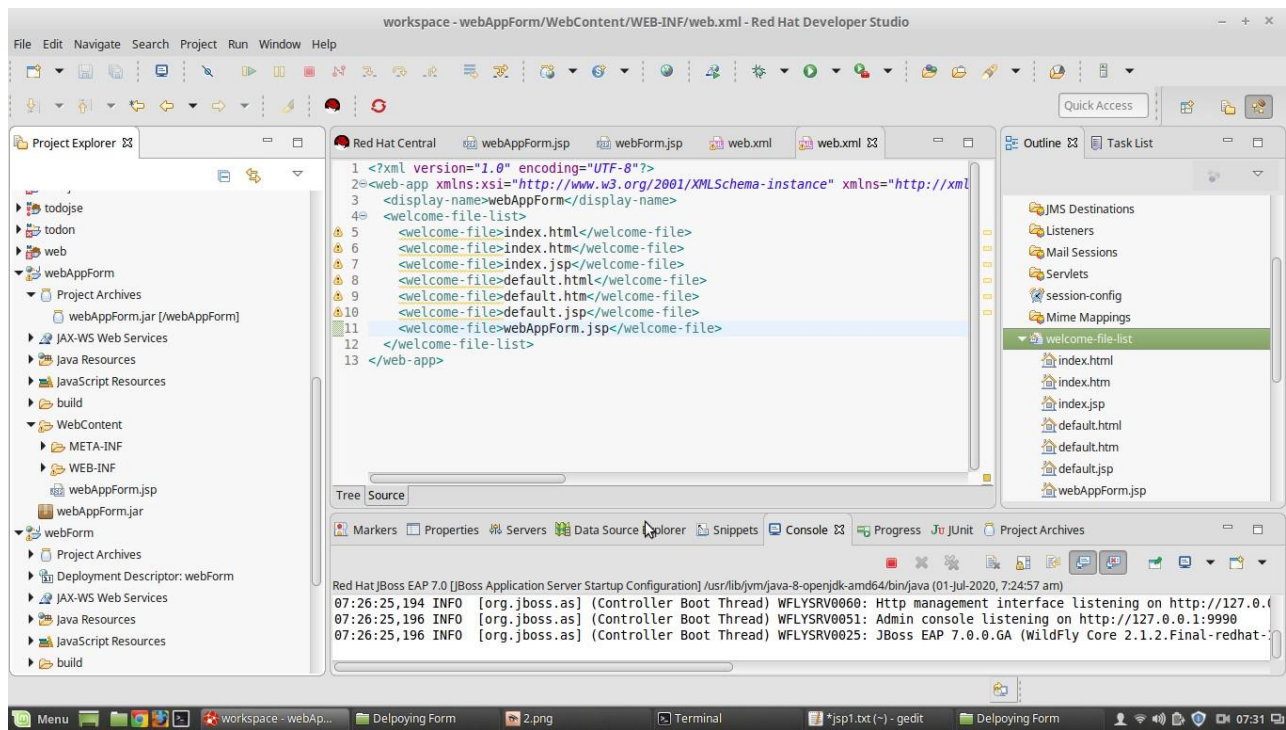




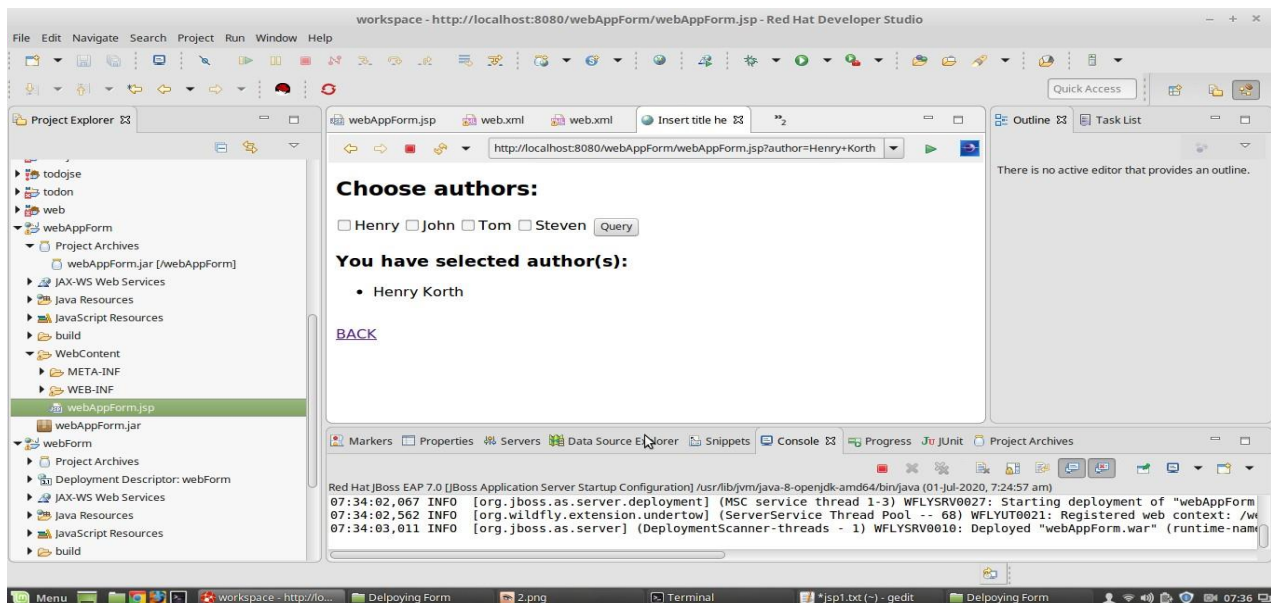
12) Select Red Hat EAP 7.0 Server. Click Finish. Server is Started



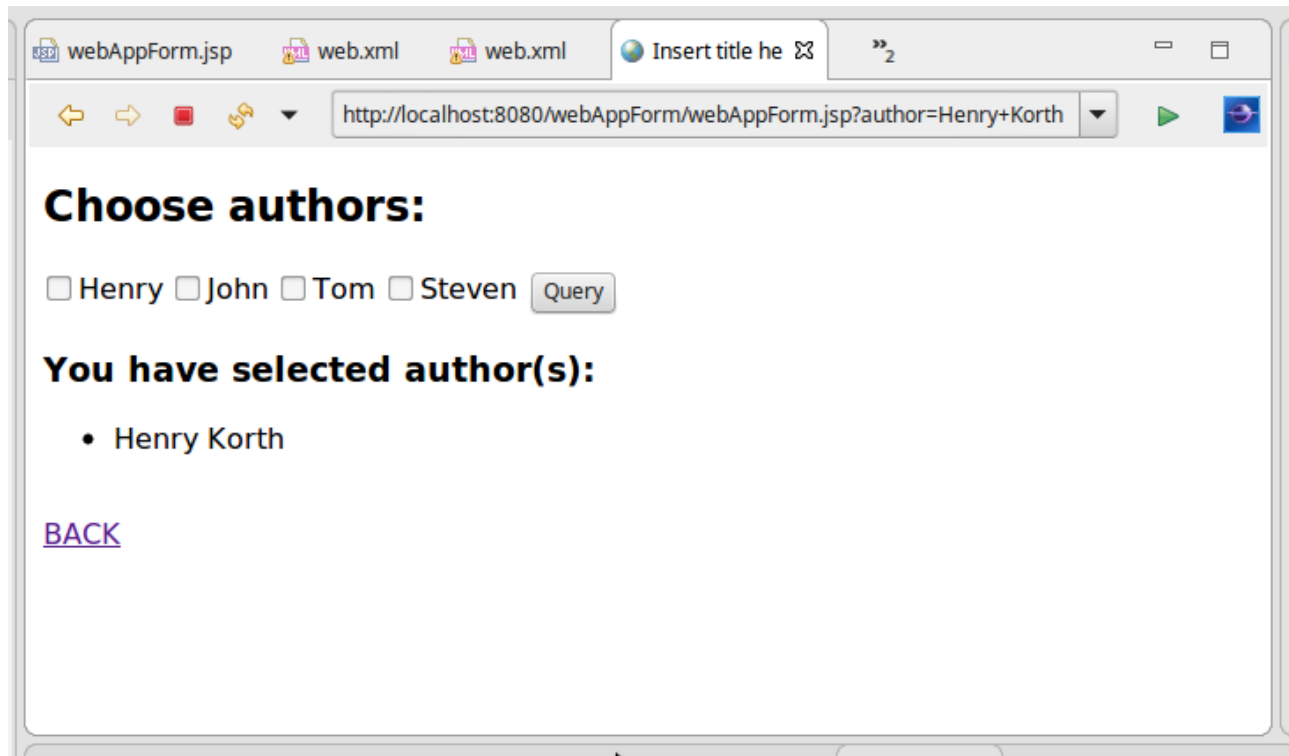
13) Right Click webAppForm.jsp → Run As → 1 Run on Server. File is deployed on Red Hat JBOSS EAP 7.0 Server webAPPForm.war File is Generated. URL <http://localhost:8080/webAppForm.jsp>



14) Select Henry Click Query



### **15) Output:-**



**Conclusion:-** In this Practical I learn Deploying web page on Red Hat JBoss EAP Server 7.0

Godavari Foundation's  
**Godavari College of Engineering, Jalgaon**  
**Department of Computer**  
**Lab Manual**  
**Database System Laboratory**  
**Practical No:- 10**

**Date:-** 01-12-2020

**Name of Student:-** Shweta Ravindra Patil.

**Class:-** T. Y. Computer Engineering.

**Roll No:-** 34

**Title:-** Large objects – CLOB, NCLOB, BLOB and BFILE.

**Aim:-** Create CLOB, NCLOB, BLOB, BFILE Datatype in a table.

**Software Requirement:-** MYSQL.

**Hardware Requirement:-** 2GB RAM.

**Theory:-**

**Types of Large Objects(LOBs)**

There are different kinds of LOBs which can be stored either in the database or in external files.

**Internal LOBs**

LOBs in the database are stored in a way that optimizes the space and provides efficient access within the database table spaces. Internal LOBs (BLOBs, CLOBs, NCLOBs) also provide transactional support (Commit, Rollback, and so on) of the database server.

- **BLOBs (Binary LOBs)** used to store unstructured binary (also called “raw”) data, such as video clips.
- **CLOBs (Character LOBs)** used to store large blocks of character data from the database character set.
- **NCLOBs (National Character LOBs)** used to store large blocks of character data from the National Character Set.

## Persistent and Temporary LOBs

Internal LOBs can be either persistent or temporary. A persistent LOB is an instance of LOB that exists in a table row in the database. A temporary LOB instance is created when you instantiate a LOB only within the scope of your local application.

A temporary instance would become a persistent instance when you insert the instance into a table row. Persistent LOBs use copy semantics method and also participate in database transactions. You can also recover persistent LOB in any events of transaction or system failure & could be easily committed or rolled back. In other words, as per ACID property that pertains to use database objects pertain to use persistent LOBs.

## BLOB

A BLOB column stores actual binary strings or byte strings in a MySQL database instead of a file path reference. This has one advantage; when you back up your database, your entire application data is copied over.

The BLOB data type is used to store large amounts of binary data. BLOB values can be converted to VARBINARY.

## BLOB Data Type

MySQL supports 4 types of BLOB data types, which only differ in the maximum length of data they can store. Here is a summary of the different types:

21. **TINYBLOB:** Only supports up to 255 bytes.
22. **BLOB:** Can handle up to 65,535 bytes of data.
23. **MEDIUMBLOB:** The maximum length supported is 16,777,215 bytes.
24. **LONGBLOB:** Stores up to 4,294,967,295 bytes of data.
- 25.

## CLOB

The CLOB data type is used to store large amounts of 7-bit ASCII character data. CLOB values can be converted to VARCHAR.

## NCLOB

The NCLOB data type is used to store a large Unicode character object. NCLOB values can be converted to NVARCHAR.

## External LOBs and the BFILE Datatype

External LOBs are data objects stored in operating system files outside the database table spaces, that have no transactional support from the database server.

BFILES are having read-only data types. The database allows read-only byte stream access to data stored in BFILES. You cannot write to a BFILE from within your application.

The database uses reference semantics with BFILE columns. Data stored in a table column of type BFILE is physically located in an operating system file, not in the database table space.

### **BFILEs basically used to hold:**

1. Binary data, which does not change while your application is running, such as graphics.
2. Data that is loaded into other large object types, such as a CLOB or BLOB where the data can be manipulated.
3. Data which is appropriate for byte-stream access, such as multimedia.
4. Read-only data which is relatively large in size, so that it will avoid taking up large amounts database table space.

Any storage device accessed by the operating system can hold BFILE data, including hard disk drives, CD-ROMs, CDs and DVDs. The database can access BFILEs provided the operating system supports stream-mode access to the operating system files.

### **Security for BFILEs**

Basically, a DIRECTORY object is used to access and use BFILEs. The DIRECTORY is an alias name for the actual physical directory in the server file system containing the file. Users are permitted to access the file only if authorized on the DIRECTORY object.

5. The DDL (Data Definition Language) SQL statements like CREATE, REPLACE, ALTER, and DROP are used with DIRECTORY database objects.
6. The DML (Data Management Language) SQL statements are used to GRANT and REVOKE object privileges on DIRECTORY objects.
7. Up to 10 BFILEs can be opened simultaneously in one session.

<b>LARGE OBJECTS DATA TYPES</b>	<b>DESCRIPTION</b>
Binary Large Object (BLOB)	Stores any kind of data in binary format such as images, audio, and video.
Character Large Object (CLOB)	Stores string data in the database having character set format. Used for large set of characters/strings or documents that use the database character.
National Character Large Object (NCLOB)	Stores string data in National Character Set format. Used for large set of characters/strings or documents in the National Character Set. Supports characters of varying width format.
External Binary File (BFILE)	BFILEs can be accessed from your application on a read-only basis. Use BFILEs to store static data, such as image data, that does not need to be manipulated in applications.

### **MYSQL QUERIES**

Create table blobex(image blob); Create table clobex(image1 clob); Create table nclobex(image2 nclob); Create table bfileex(imgFile bfile);

**Conclusion:-** In this Practical I learn CLOB, NCLOB, BLOB, BFILE Datatype in a table.

Godavari Foundation's  
Godavari College of Engineering, Jalgaon  
Department of Computer  
**Lab Manual**  
Database System Laboratory  
**Practical No:- 11**

**Date:-** 04-12-2020

**Name of Student:-** Shweta Ravindra Patil.

**Class:-** T. Y. Computer Engineering.

**Roll No:-** 34

**Title:-** Distributed data base Management, creating web-page interfaces for database applications using servlet.

**Aim:** - Create web-pages in HTML and insert record in MYSQL database using servlet. Execute it on Red Hat JBoss EAP Server 7.0

**Software Requirement:-** MYSQL , HTML , Servlet , Redhat JBOSS EAP Server 7.0.

**Hardware Requirement:-** 2GB RAM.

**Theory:-**

**HTML and MYSQL Database Connectivity using Sevlet**



## 1) Start MYSQL . Create Table in Student Database.

```
Terminal
File Edit View Search Terminal Help
Server version: 5.5.62-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use student;
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> desc t;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id     | int(11)       | YES  |     | NULL    |       |
| name   | varchar(30)   | YES  |     | NULL    |       |
| emailid | varchar(30)   | YES  |     | NULL    |       |
| country | varchar(30)   | YES  |     | NULL    |       |
| gender | varchar(20)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.28 sec)

mysql>
```

## 2) Shows Structure of Table by desc; and also Select \* from t; Then Empty set message is displayed.

```
Terminal
File Edit View Search Terminal Help
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use student;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

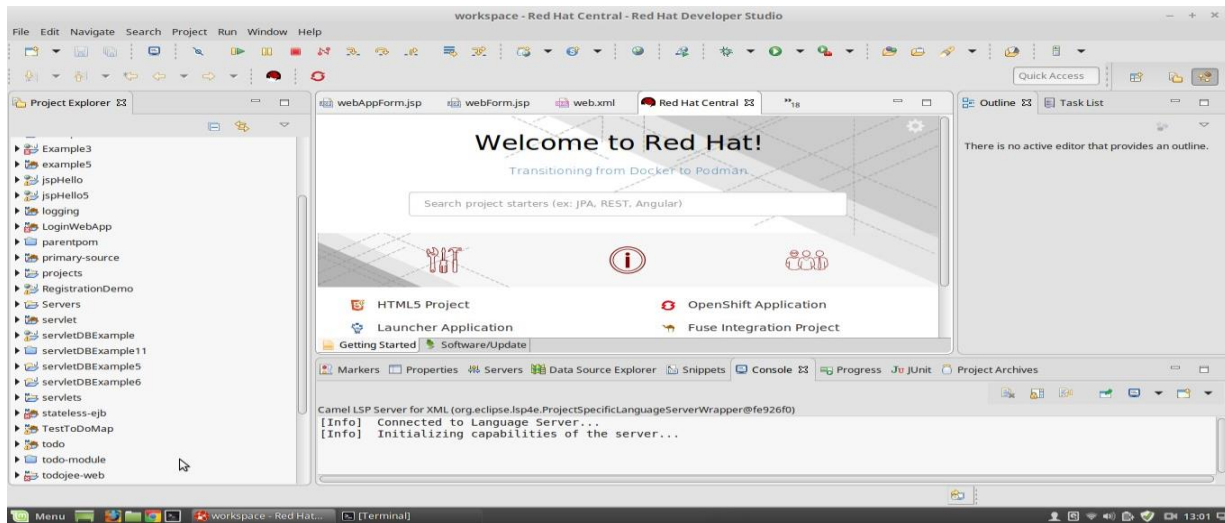
Database changed
mysql> desc t;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id     | int(11)       | YES  |     | NULL    |       |
| name   | varchar(30)   | YES  |     | NULL    |       |
| emailid | varchar(30)   | YES  |     | NULL    |       |
| country | varchar(30)   | YES  |     | NULL    |       |
| gender | varchar(20)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.28 sec)

mysql> delete from t;
Query OK, 1 row affected (0.83 sec)

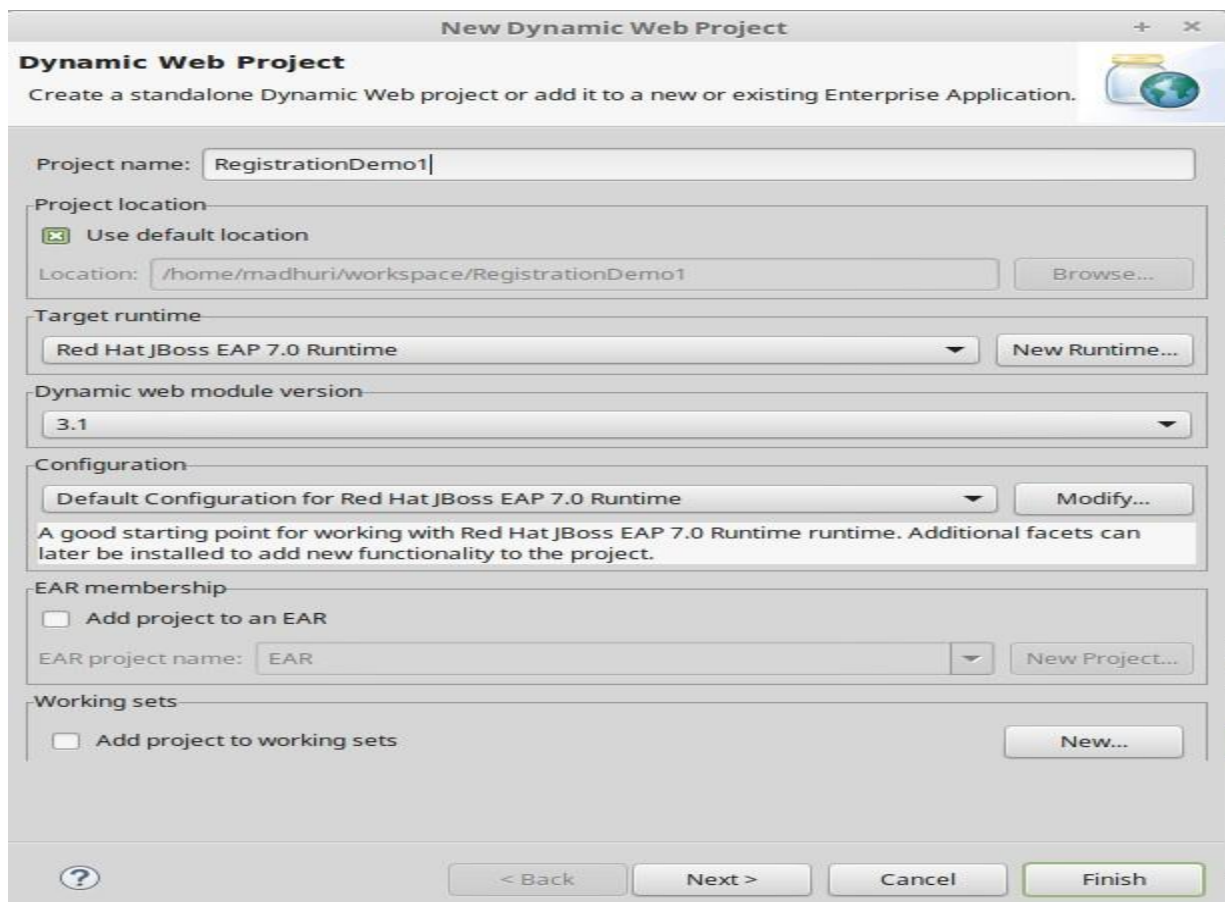
mysql> select * from t;
Empty set (0.08 sec)

mysql>
```

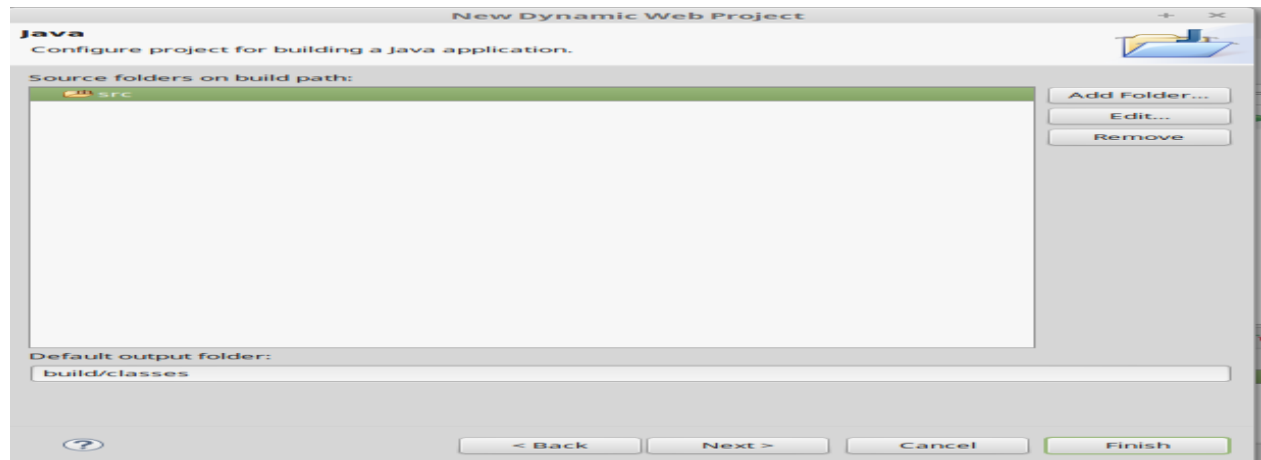
### 3) Open Red Hat Developer Studio EAP 7.0



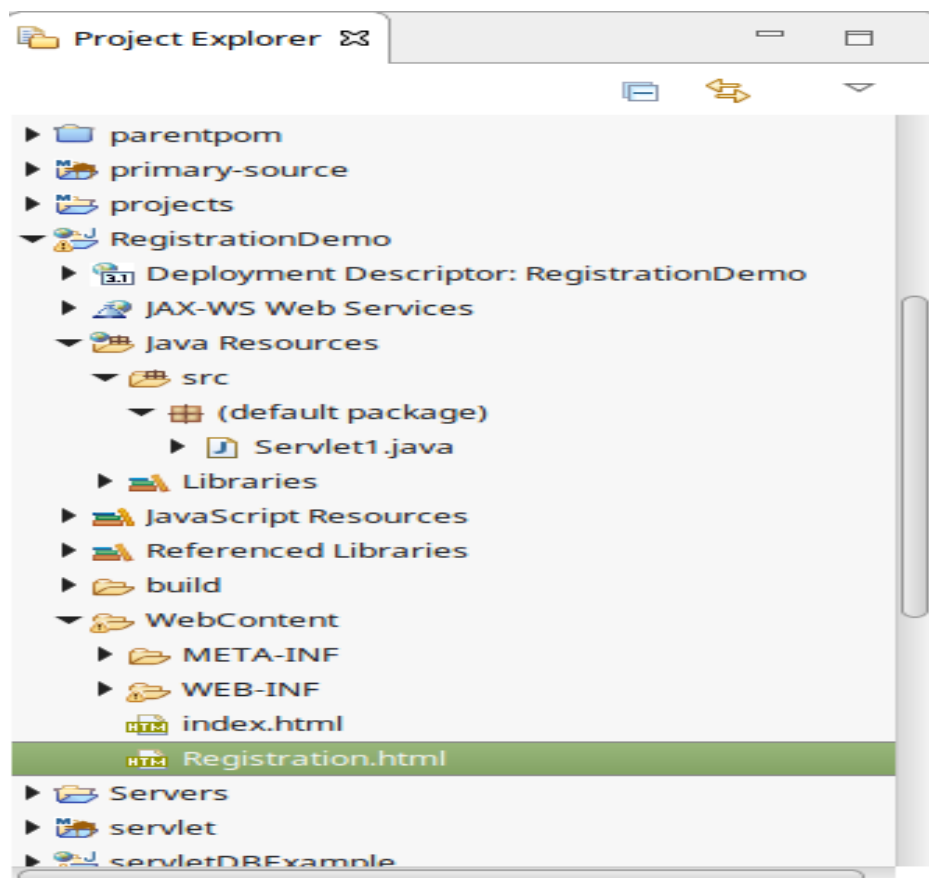
### 4) Click File → New → Dynamic Web. Click Project & Create New Project RegistrationDemo.



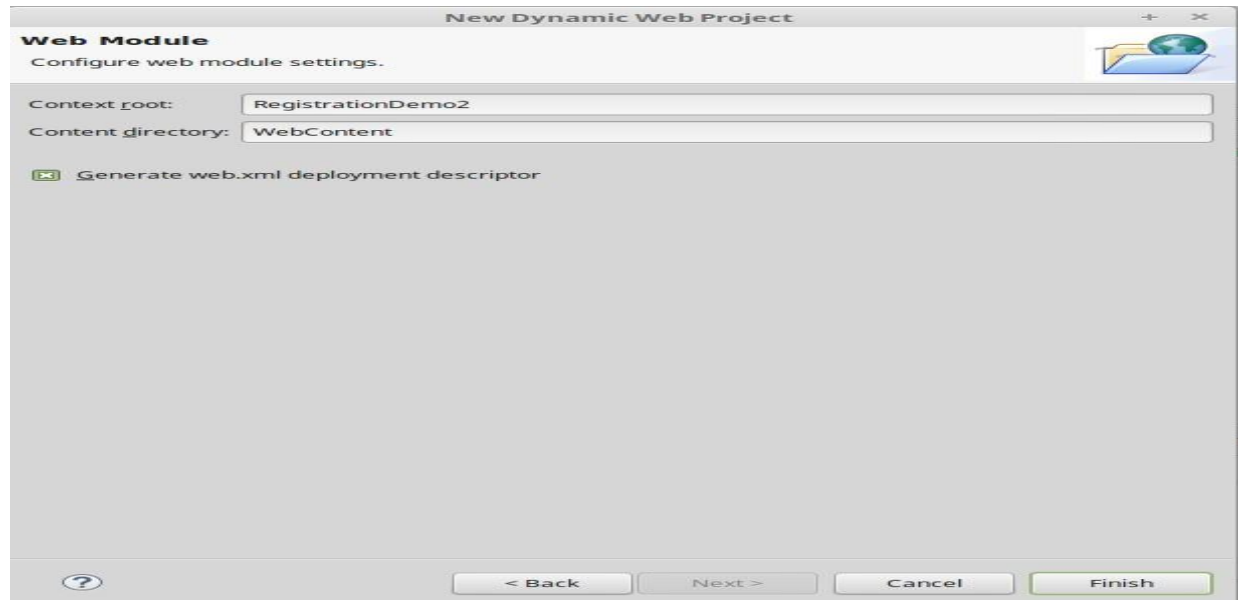
5) Click Next ->



6) Click Next → select Generate web.xml deployment descriptor. Create 3 Files index.html , Registration.html and Servlet1.java file



7) Right Click WebContent → New → HTML → index.html . Copy and Paste Code. With Respective File.



```
<!DOCTYPE html>
```

```
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<a href="Registration.html">Registration</a>
</body>
</html>
```

8) Right Click WebContent → New → HTML → Registration.html Copy and Paste Code.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<div style="height:300px;width:300px;margin-top:200px;margin-left:300px;border-
color:red;border-style:solid;border-width:4px;">
<h1>Registration</h1>
<form action="Servlet1">
<table>
<tr>
<td>Enter the ID:</td>
<td><input type="text" name="id"></td>
</tr>
<tr>
<td>Enter the Name:</td>
```

```

<td><input type="text" name="name"></td>
</tr>
<tr>
<td>Enter the Email ID:</td>
<td><input type="email" name="emailid"></td>
</tr>
<tr>
<td>Select the Country:</td>
<td><select name="country">
<option>India</option>
<option>America</option>
<option>England</option>
</select>
</td>
</tr>
<tr>
<td>Select the Gender:</td>
<td><input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="female">Female
</td>
</tr>
<tr>
<td>Submit Information:</td>
<td><input type="submit" value="submit"></td>
</tr>
<tr>
</table>
</form>
</div>
</body>
</html>

```

9) Right Click src → New → Servlet → Servlet1.java Copy and Paste Code.

```

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException; import
javax.servlet.annotation.WebServlet; import
javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;
/* Servlet implementation class Servlet1 */ @WebServlet("/Servlet1")

public class Servlet1 extends HttpServlet {

    private static final long serialVersionUID = 1L;

```

```

/* @see HttpServlet#HttpServlet() */

public Servlet1() {
    super();
    // TODO Auto-generated constructor stub
}

/*@see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response) */

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    // TODO Auto-generated method stub
    response.getWriter().append("Served at: ").append(request.getContextPath());

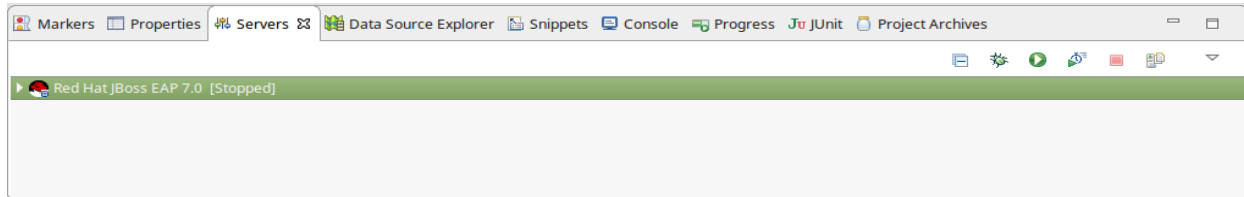
    PrintWriter out=response.getWriter();
    String id=request.getParameter("id");
    String name=request.getParameter("name");
    String emailid=request.getParameter("emailid");
    String country=request.getParameter("country");
    String gender=request.getParameter("gender");

    // jdbc code for connectivity with mysql try
    {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/student","root","root");
        Statement stmt = con.createStatement();
        stmt.executeUpdate("insert into t
values("+id+", '"+name+"', '"+emailid+"', '"+country+"', '"+gender+"')");
        out.print(" Your Record has been successfully inserted");
    }
    catch(Exception p){ }
}

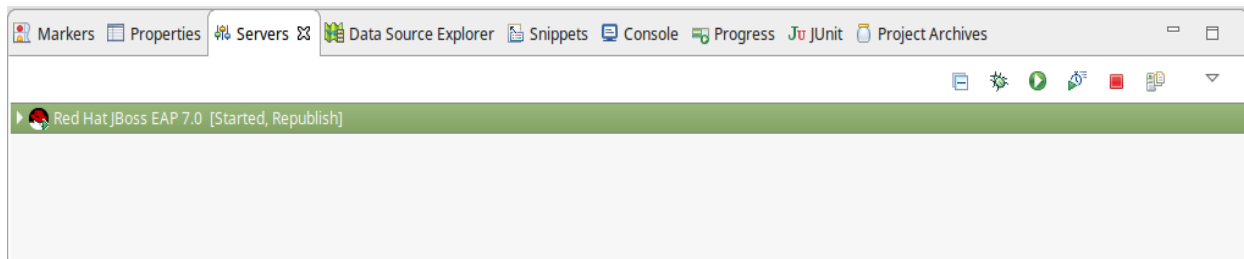
/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    // TODO Auto-generated method stub
    doGet(request, response);}

```

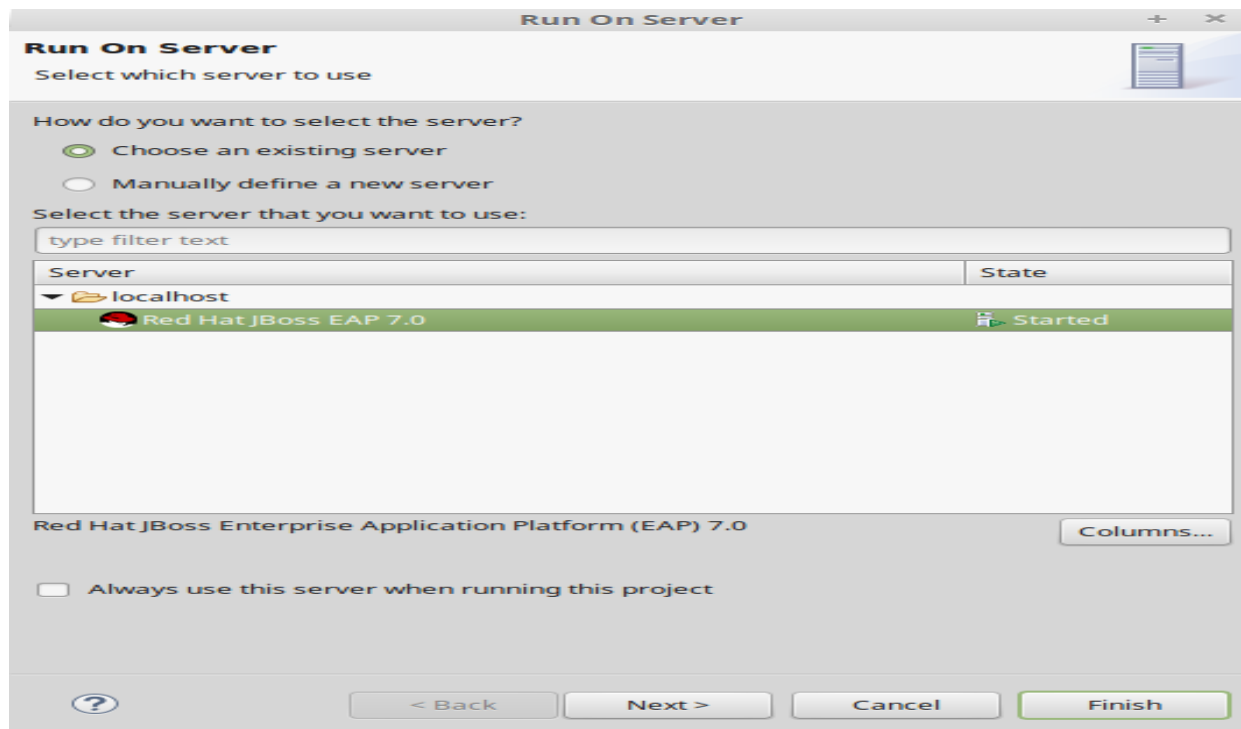
10) Click Servers Tab



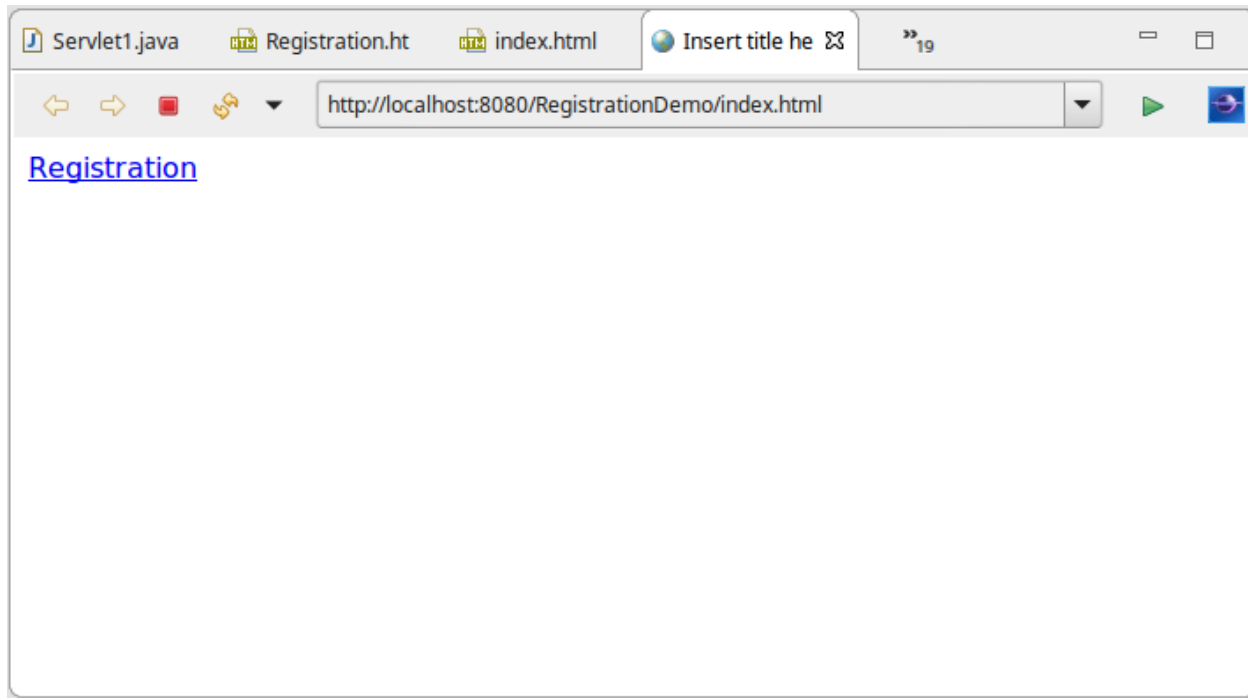
11) Right Click Red Hat JBoss EAP 7.0 [Stopped] → Start



12) Select Red Hat JBOSS EAP 7.0 Click Next → Click Finish. Server is Started.



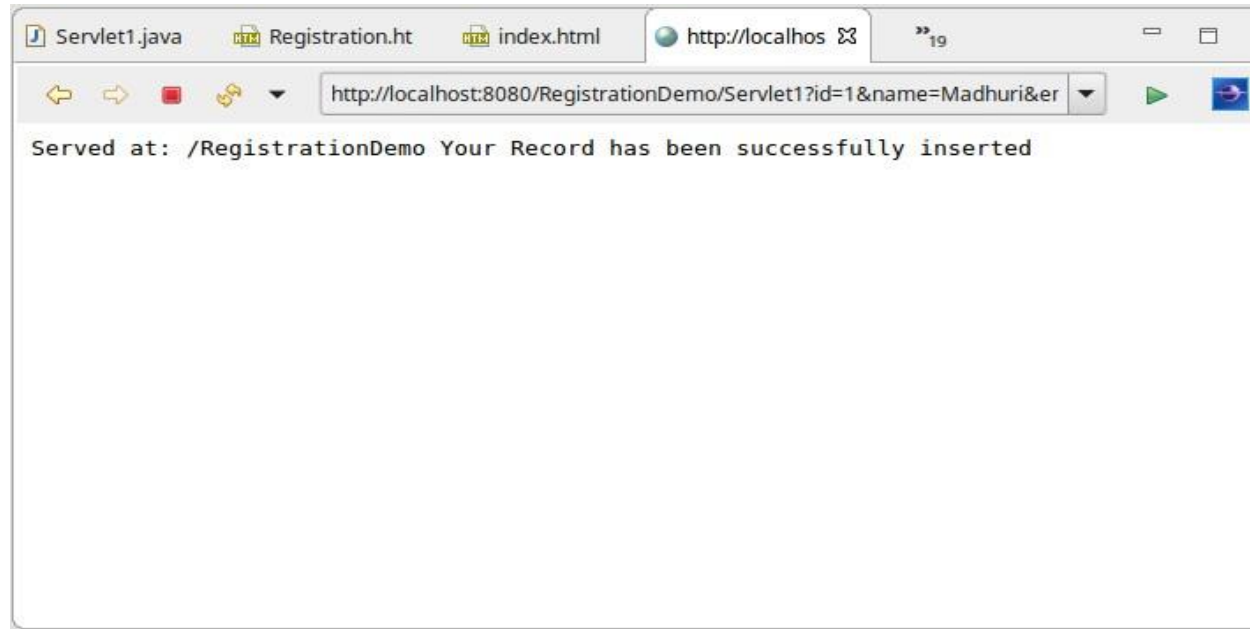
13) Right Click index.html → Run As → 1 Run on Server Click Registration.



14) Fill Information in form. Click Submit.

A screenshot of a web browser window displaying a registration form. The address bar shows the URL 'http://localhost:8080/RegistrationDemo/Registration.html'. The form is titled 'Registration' in a large, bold, black font. Below the title, there are several input fields and a submit button, all enclosed within a red rectangular border. The form fields are: 'Enter the ID:' with a text box containing '1'; 'Enter the Name:' with a text box containing 'Madhuri'; 'Enter the Email ID:' with a text box containing 'zawarmadhuri@gmail.com'; 'Select the Country:' with a dropdown menu showing 'India'; 'Select the Gender:' with radio buttons for 'Male' and 'Female', where 'Female' is selected; and a 'Submit' button labeled 'submit'.





15 ) Check Database by command : **select \* from t;** in MYSQL.  
It's Shows that Row is inserted in Table in MYSQL.

```
Terminal
File Edit View Search Terminal Help
Database changed
mysql> desc t;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id     | int(11) | YES | | NULL | |
| name   | varchar(30) | YES | | NULL | |
| emailid | varchar(30) | YES | | NULL | |
| country | varchar(30) | YES | | NULL | |
| gender | varchar(20) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.28 sec)

mysql> delete from t;
Query OK, 1 row affected (0.83 sec)

mysql> select * from t;
Empty set (0.08 sec)

mysql> select * from t;
+-----+-----+-----+-----+-----+-----+
| id | name | emailid | country | gender |
+-----+-----+-----+-----+-----+-----+
| 1 | Madhuri | zawarmadhuri@gmail.com | India | female |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

**Conclusion:-** In this Practical I learn to create web-pages in HTML and insert record in MYSQL database using servlet. Execute it on Red Hat JBoss EAP Server 7.0