INDEX

|  |  |  |
| --- | --- | --- |
| **SR. NO.** | **TOPIC** | PAGE NO. |
| 1 | ABSTRACT | 2 |
| 2 | REQUIREMENT SPECIFICATION | 4 |
| 3 | ER DIAGRAM | 6 |
| 4 | TABLE DESCRIPTION | 7 |
| i | SQL COMMANDS | 9 |
| ii | BASIC COMMANDS | 11 |
| iii | CONSTRAINTS | 12 |
| iv | SELECT STATEMENT | 14 |
| v | RESTRICTING AND SORTING DATA | 15 |
| vi | AGGREGATION | 17 |
| vii | SINGLE ROW FUNCTIONS | 18 |
| viii | SUBQUERIES | 20 |
| ix | DISPLAYING DATA FROM MULIPLE  TABLES | 22 |
| x | VIEWS | 27 |
| xi | NORMALISATION | 29 |

**ABSTRACT**

This project contains Introduction to the Railways reservation system. It is the computerized system of reserving the seats of train in advanced. It is mainly used for long route. Online reservation has made the process for the reservation of seats very much easier than ever before. In our country India, there are number of counters for the reservation of the seats and one can easily make reservations.

**RAILWAY MANAGEMENT SYSTEM**

Railway management System is the computerized system of reserving the seats of train seats in advanced . It is mainly used for long route . On-line reservation has made the process for the reservation of seats very much easier than ever before. In our country India, there are number of counters for the reservation of the seats and one can easily make reservations and get tickets. Almost all of the manual reservation centres have been converted into computerized reservation centers, which provide booking.

This system is basically concerned with the reservation and cancellation of railway tickets to the passengers. The need of this system arose because as is the known fact that India has the largest railway network in the whole of the world and to handle it manually is quite a tough job. By computerizing it, we will be able to overcome many of its limitations and will be able to make it more efficient. The handling of data and records for such a vast system is a very complex task if done manually but it can be made much easier if the system is computerized. To be more specific, our system is limited in such a way that a train starting from a source will have a single destination.

The basic functions being performed by our system are:

1. RESERVATION MANAGEMENT
2. FARE MANAGEMENT
3. TIMETABLE MANAGEMENT

These functions will be handled with the help of following sub functions: -

 It reserves and cancels seats for the passenger.

 It contains information about the trains.

 It contains information about the passenger.

 It contains the details of reservation fees, any concessions etc.

 It makes entries for reservation, waiting, cancelled tickets.

 It will update for uptime and downtime trains.

A railway system, which needs to model the following:

1. Stations
2. Tracks, connecting stations. You can assume for simplicity that only one track exists between any two stations. All the tracks put together form a graph.
3. Trains, with an ID and a name
4. Train schedules recording what time a train passes through each station on its route.

You can assume for simplicity that each train reaches its destination on the same day, and that every train runs every day. Also for simplicity, assume that for each train, for each station on its route,  you store

* Time in,
* Time out (same as time in if it does not stop)
* A sequence number so the stations in the route of a train can be ordered by sequence number.

**2: REQUIREMENT SPECIFICATION**

**PROJECT PERSPECTIVE**

This project represents the initial version of the Railway management system. All requirements listed herein describe a self-contained system. At a high level, this project will allow a user to book tickets, check ftrains, do account maintenance, and query train information. The goal is to allow customers greater and easier access to the railway booking system, twenty-four hours a day.

**MODULES :**

**Login**

**Description:**

 This function allows a registered user to login his account using his frequent travelling number with the airline and password. If a user is not registered, the website shall allow the user to enrol first. The system will check both the frequent train number and password, when a user attempts to login. This provides security to the system by authenticating each memberand provides confidence to the consumer that his/her personal information is secure.

**Enrolment**

**Description:**

This function allows unregistered user to enrol and to create a new account with the website. In order to create a new account, the user has to provide required information such as first name, last name, email address and password. Other optional information, such as phone number, credit card information and mailing address, can be provided during the registration process. The system checks if all required data are provided and then will prompt the user to enter additional information, if required. After all required information is provided, the system auto-generates a unique frequent travelling number that the user must use as username for future authentications. The system shall auto-generate this number in less than five seconds.

 A user who wishes to purchase flights and use advanced features,must be logged in. However, without enrolment, a user can never be member. This section offers all users a chance to become a member.

**Book Trains**

**Description:**

The user can use the Book trains function to purchase seats for an railway train. The system shall present the user with information on all current trains. The user may then select a pair (departure and return) of trains on which to purchase seats. The user can indicate the number of seats and placement of such. Finally, the system shall guide the user completely through the checkout process.

The heart of the business is selling seats on trains. This sectionprovides the primary source of system transactions.

**Reserve Seats**

**Description:**

The user can use the Reserve Seatfunction to reserve seats for a train. The seats to be reserved are initially found through the user’s previous bookings. These bookings were previously completed through the

 Book train  function .The system shall display available seats for the departing and returning trains booked by the user. The user selects seats from each train, where the number of selected seats from each train is the number that the user booked on that particular train. Once the train seats are selected, the user confirms the seat selection.

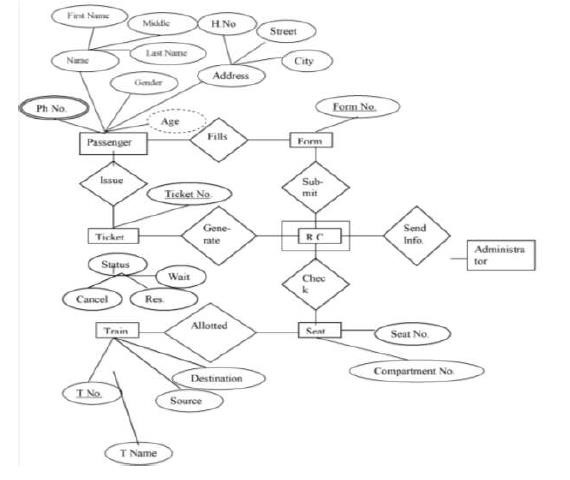
:Customers prefer to know where their seats are located. Further, theyprefer to pick out particular seats – closer to the front, window seat, middle berth, etc. From that point, the seats are removed from available/unreserved seats and the user’s booking is linked to those particular seats. If the user fails to reserve a seat prior to train departure the user is randomly assigned a seat from available seats a 30 minutes prior to initial take-off time. This function is offered immediately after booking the user can wait and use the function to book seats anytime after up until 30minutes prior their flight

**Train Status**

**Description:**

This section shall allow the user – whether enrolled or not – to view flight information that matches input criteria. The user will provide:1.A flight number and DateOR2.Departing/Arriving Cities and Date. The system will display matching train information including the following fields

**ER DIAGRAM**



**Cardinality**

Cardinality notations

M = many, N = 0, 1, 2 …

**TABLE DESCRIPTION**

There are 4 tables in the project:-

i. Passenger ii. Ticket iii. Form iv. Seat

Passenger table contains the ID from all the tables and other table use this ID as foreign key referring to Passenger.

Desc Train:

Name Null? Type

----------------------------------------- -------- ----------------------------

TRAIN\_ID NOT NULL NUMBER(38)

TRAIN\_NAME VARCHAR(50)

SOURCE VARCHAR(50)

DESTINATION VARCHAR(50)

Desc Train\_info:

Name Null? Type

----------------------------------------- -------- ----------------------------

TRAIN\_NO NUMBER(38)

SLEEPER\_SEATS NUMBER(38)

1AC\_SEATS NUMBER(38)

2AC\_SEATS NUMBER(38)

3AC\_SEATS NUMBER(38)

FARE NUMBER(38)

DATE DATE

Desc Coach:

Name Null? Type

----------------------------------------- -------- ----------------------------

TRAIN\_NO NUMBER(38)

COACH NUMBER(38)

SEATS NUMBER(38)

Desc Passenger:

Name Null? Type

----------------------------------------- -------- ----------------------------

PASS\_NO NUMBER(38)

PASS\_NAME VARCHAR2(25)

TRAIN\_NO NUMBER(38)

TRAIN\_NAME VARCHAR2(50)

SOURCE VARCHAR2(50)

DESTINATION VARCHAR2(50)

COACH\_NO NUMBER(38)

SEAT NUMBER(38)

AGE NUMBER(10)

SEX VARCHAR2(50)

## SQL COMMANDS

### BASIC COMMANDS

a) TABLE CREATION:

create table Train

(Train\_no int,

Train\_name varchar (50),

Source varchar (50),

Destination varchar(50),

);

Table Created

create table Train Info (Train\_no int,

Sleeper\_Seats int,

1ac seats int,

2ac seats int,

3ac seats int,

Fare int,

Date date,

);

Table created.

create table Coach ( Train\_no int,

Coach int,

Seats int,

);

Table created.

CREATE TABLE Passenger

( Pass\_no int,

Pass\_name varchar(50),

Pnr\_no int,

Train\_no. varchar(50),

Train\_name varchar(50),

Source varchar(50),

Destination varchar(50),

Coach\_no int,

Seat int,

Age int,

Sex varchar(50)

);

#### b) INSERTION OF VALUES

SQL> insert into Train values(001,Rajdhani,Delhi,Patna); 1 row created.

SQL> insert into Train values(002,Shatabdi,Ahmedabad,Mumbai);

1 row created.

SQL> insert into Train values(003,Chennai Express,Chennai,Banglore); 1 row created.

SQL> insert into Train\_info values(001,200,20,80,180,2000,12/10/2018);

1 row created.

SQL> insert into Coach values(001,S5,24); 1 row created.

SQL> insert into Passenger values(1,Shatakshi,24JK8970,001,Rajdhani,Delhi,Patna,S5,24); 1 row created.

SQL> insert into Passenger values(2,Pranav,30GMC691,002,Ahmedabad,Mumbai,B2,68); 1 row created.

1. MODIFY

SQL> alter table Train\_info

Modify Date int;

Table altered.

1. TRUNCATE, DROP

SQL> truncate table Train; Table truncated. SQL> drop table Coach ;

Table dropped.

### CONSTRAINTS

a) PRIMARY KEY and FOREIGN KEY

create table Train (Train\_no int primary key,

Train\_name varchar (50),

Source varchar (50),

Destination varchar(50),

);

Table Created

create table Train Info

(Train\_no int Primary key,

Sleeper\_Seats int,

1ac seats int,

2ac seats int,

3ac seats int,

Fare int,

Date date,

);

Table created.

create table Coach

( Train\_no int Primary Key,

Coach int,

Seats int,

);

Table created.

CREATE TABLE Passenger

( Pass\_no int Primary Key,

Pass\_name varchar(50),

Pnr\_no int,

Train\_no. varchar(50),

Train\_name varchar(50),

Source varchar(50),

Destination varchar(50),

Coach\_no int,

Seat int,

Age int,

Sex varchar(50)

);

1. CHECK CONSTRAINT

SQL> alter table Passenger add check(Age>12);

1. NOT NULL CONSTRAINT

create table Train

(Train\_no int primary key NOT NULL,

Train\_name varchar (50) NOT NULL,

Source varchar (50),

Destination varchar(50),

);

Table Created

1. DEFAULT CONSTRAINT

SQL> alter table Train modify Source default 'Delhi';

1. UNIQUE CONSTRAINT

SQL> alter table Passenger add unique(Pnr\_no);

### SELECT STATEMENTS

#### SELECTING ALL COLUMNS

SQL> select \* from Train;

TRAIN\_NO TRAIN\_NAME

---------- --------------- 001 Rajdhani

002 Shatabdi

003 Chennai Express

#### RENAMING OF TABLE NAME

SQL> alter table Train\_Info rename to Train\_Information; Table altered.

#### c) ARITHMETIC EXPRESSION

SQL> select Train\_no,(S5-Coach\_no) Coach\_no from Coach;

TRAIN\_NO COACH\_NO

---------- ----------

001 S5

002 S5

003 S5

#### d) DEFINING NULL VALUES

SQL> select \* from Train where Train\_no is not null;

TRAIN\_NO TRAIN\_NAME SOURCE DESTINATION ---------- ------------------------- ------------------------- ---------- ----------

001 Rajdhani Delhi Patna

#### USING COLUMN ALIASE

SQL> select Train\_no,Train\_name as Train\_Details from Passenger;

TRAIN\_NO TRAIN\_NAME

---------- ----------

003 Chennai Express

#### USING CONACTENATION, COLUMN ALIASE

SQL> select Train\_no ||' is starting from '||source as Journey from Train;

TRAIN

--------------------------------------------------------------------------------

001 is starting from Delhi

002 is starting from Ahmedabad

003 is starting from Chennai

### RESTRICTING AND SORTING DATA

#### USING WHERE CLAUSE

SQL> select Train\_name from Train where source='Chennai';

TRAIN\_NAME

-------------------------

Chennai Express

#### b) USING LIKE

SQL> select Train\_name from Train where destination like 'mu%';

TRAIN\_NAME

-------------------------

Shatabdi

#### c) USING AND, OR, NOT

SQL> select \* from Passenger where Pass\_no ='1' and Source like 'de%';

PASS\_NO PASS\_NAME PNR\_NO TRAIN\_NO TRAIN\_NAME SOURCE

---------- ------------------------- ------------------------- ---------- ---------- 1 Shatakshi 24JK8970 001 Rajdhani Delhi

SQL> select \* from Coach where Train\_no=001 or Train\_no=003;

TRAIN\_NO COACH SEAT

---------- ---------------

001 S5 24

003 B12 75

#### d) ORDERBY CLAUSE

SQL> select \* from Train\_Info order by date;

TRAIN\_NO SLEEPER SEATS 1AC 2AC 3AC FARE DATE

---------- ---------------

001 250 10 80 180 2000 12/10/2018 1 row selected.

### AGGREGATING OF DATA USING GROUP FUNCTIONS

#### a) AVG AND SUM

SQL> select avg(date) from Train\_info;

AVG(DATE)

-------------------------

17.3333333

#### b) MIN AND MAX

SQL> select min(date), max(date) from Train\_info;

MIN(DATE) MAX(DATE)

-------------------- -------------------- 10/08/2018 21/11/2018

#### COUNT

#### 

SQL> select count(\*) from Train;

COUNT(\*)

----------

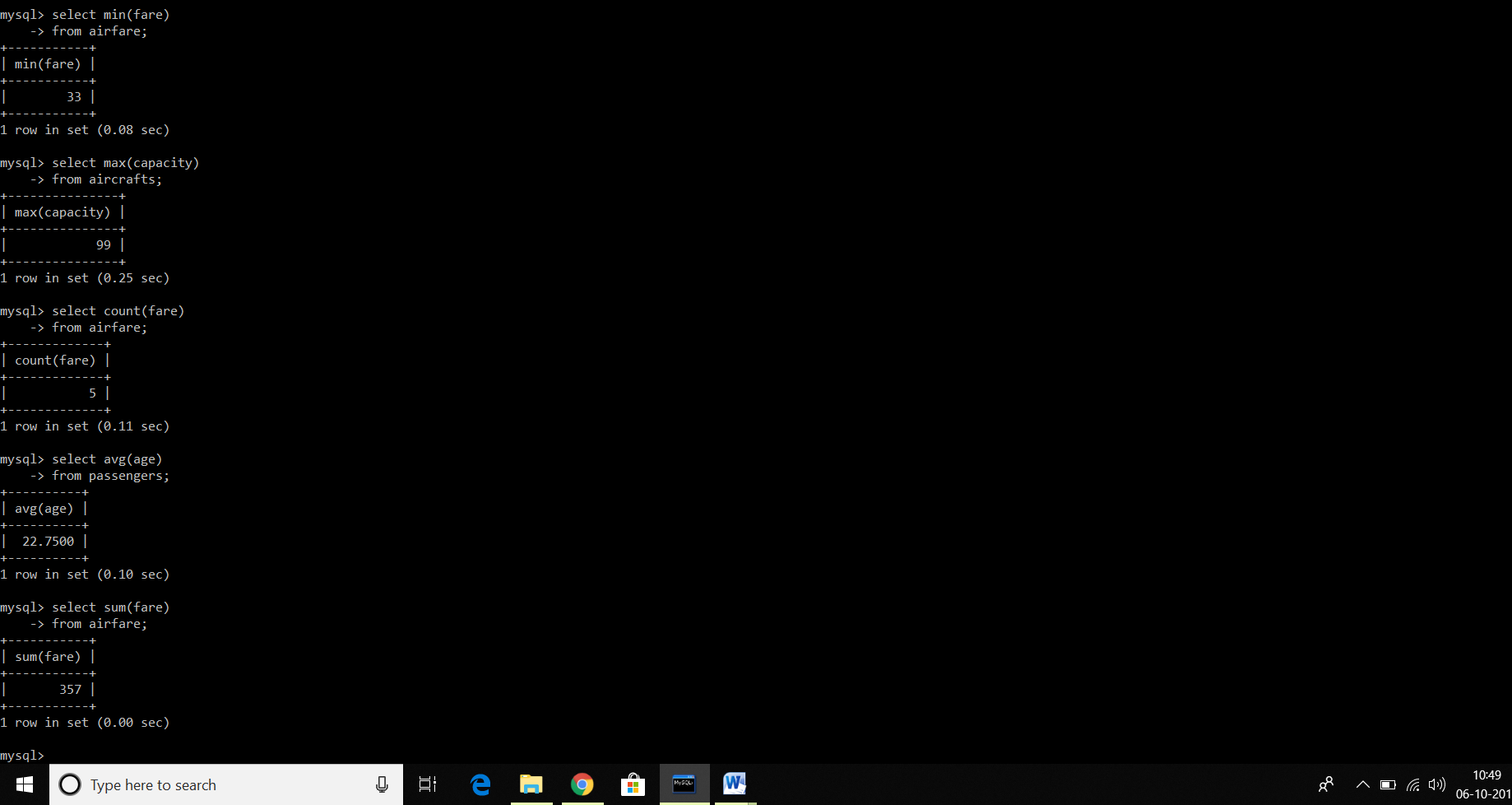
3

#### DISTINCT

SQL> select count(distinct Train\_id) from Train;

TRAIN(DISTINCT TRAIN\_ID)

-----------------



### SINGLE ROW FUNCTIONS

a) UPPER AND LOWERSQL> select upper(Pass\_name) from Passenger;

UPPER(PASS\_NAME)

-------------------------

SHATAKSHI

SQL> select lower(Pass\_name) from Passenger;

LOWER(PASS\_NAME) -------------------------

Shatakshi

#### b) CHARACTER MANIPULATION

SQL> select concat(Train\_id, Train\_name) as details from Train;

DETAILS

-----------------------------------------------------------------

001Rajdhani

002Shatabdi

003Channai Express

SQL> select lpad(Train\_name,10) from Train;

LPAD(TRAIN\_NAME,10)

----------

Rajdhani

Shatabdi

SQL> select length(Train\_name) from Train;

LENGTH(Train\_name)

-----------

9

SQL> select trim(Train\_name) from Train;

TRIM(TRAIN\_NAME)

--------------------

Rajdhani

#### c) NUMBER FUNCTIONS

SQL> select trunc(date,1) from Train\_info;

TRUNC(DATE,1)

------------------------ 13/03/2018

05/05/2018

21/06/2018

07/09/2018 19/10/2018

5 rows selected.

SQL> select mod(Fare,3) from train\_info ;

MOD(Fare, 3)

------------------------

2

1 rows selected.

#### d) DATE AND TIME

SQL> select Date from Train\_info ;

Date

---------

12/10/2018

### SUBQUERIES

SQL> select train\_no, name

1. from Train\_info , passenger
2. where fare>(select avg(fare) from Train\_info);

TRAIN\_NO NAME ----------------- -------------------------

001 Shatakshi

002 Pranav

SQL> select train\_no , passenger.name , Train\_info.date

1. from Train\_info , passenger
2. where faculty.id=professor.prof\_id;

TRAIN\_NO NAME DATE

---------- ----------------- ---------------

001 Shatakshi 12/10/2018

002 Pranav 25/08/1998

SQL> select Train\_no

1. from Train\_info
2. where Train\_no = ANY (select Train\_no from Train\_info where Destination ='Bangalore');

Train\_no

----------

003

SQL> select Train\_no

1. from Train\_info
2. where Train\_no = ALL (select Train\_no from Train\_info where Destination = ‘Mumbai’);

Train\_no

----------

002

### DISPLAYING DATA FROM MULTIPLE TABLES

SQL> select Train\_no , passenger.pass\_name , Train\_info.Date

1. from Train\_info , passenger
2. where Train\_info.Train\_no = Passenger.Pass\_name;

TRAIN\_NO PASS\_NAME DATE

-------------- ------------------------- ---------------

001 Shatakshi 12/10/1998

002 Pranav 25/08/1998

SQL> select Train\_info.Train\_no , Passenger.Pass\_name;

1. from Train\_info
2. inner join Passenger on Train\_info.Train\_no = Passenger.Pass\_name ; TRAIN\_NO PASS\_NAME

----------------- ------------------

001 Shatakshi

003 Rahul

SQL> select Train\_info.Train\_no , Passenger.Pass\_name;

1. from Train\_info
2. full outer join passenger on Train\_info.Train\_no = Passenger.Pass\_name;

TRAIN\_NO PASS\_NAME

---------------- ------------------

001 Shatakshi

002

003 Rahul 004

1. rows selected.

SQL>select Train\_info.Train\_no , Passenger.Pass\_name;

1. from Train\_info
2. cross join passenger;

TRAIN\_NO PASS\_NAME

---------------- ----------------

001 Shatakshi

002 Shatakshi

003 Shatakshi

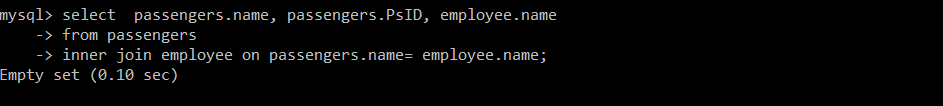
004 Shatakshi

001 Rahul

002 Rahul

003 Rahul 004 Rahul

8 rows selected.



**SQL FUNCTIONS :**

**1: CONCAT:**

The CONCAT() function adds two or more expressions together.

CONCAT(expression1, expression2, expression3,...)

1: mysql> select concat(capacity," ",MfdBy," ", MfdOn) as details

-> from aircrafts;

**2: LOWER:**

The LOWER() function converts a string to lower-case.

LOWER(text)

mysql> SELECT lower("THIS IS MySQl");

**3: REVERSE:**

The REVERSE() function reverses a string and returns the result.

REVERSE(string)

mysql> select reverse("THIS IS MYSQL");

**4: SUBSTRING:**

The SUBSTRING() function extracts some characters from a string.

SUBSTRING(string, start, number\_of\_chars)

mysql> SELECT SUBSTRING('SQL Project', 1, 3) AS ExtractString;

**5: ABS:**

The ABS() function returns the absolute value of a number.

ABS(number)

SELECT Abs(-243.5) AS AbsNum;

**6: CELING/FLOOR:**

The CEILING() function returns the smallest integer value that is >= a number.

CEILING(number)

The FLOOR() function returns the largest integer value that is <= to a number.

FLOOR(number)

1: SELECT FLOOR(25.75);

2: SELECT CEIL(25.75);

**7: CURRENT TIMESTRAMP:**

The CURRENT\_TIMESTAMP function returns the current date and time, in a 'YYYY-MM-DD hh:mm:ss.mmm' format.

CURRENT\_TIMESTAMP

SELECT CURRENT\_TIMESTAMP;

**8: DATEADD:**

DATEADD() function adds a time/date interval to a date and then returns the date.

DATEADD(interval, number, date)

SELECT DATEADD(year, 1, '2017/08/25') AS DateAdd;

**9: DATEDIFF:**

The DATEDIFF() function returns the difference between two dates.

DATEDIFF(interval, date1, date2)

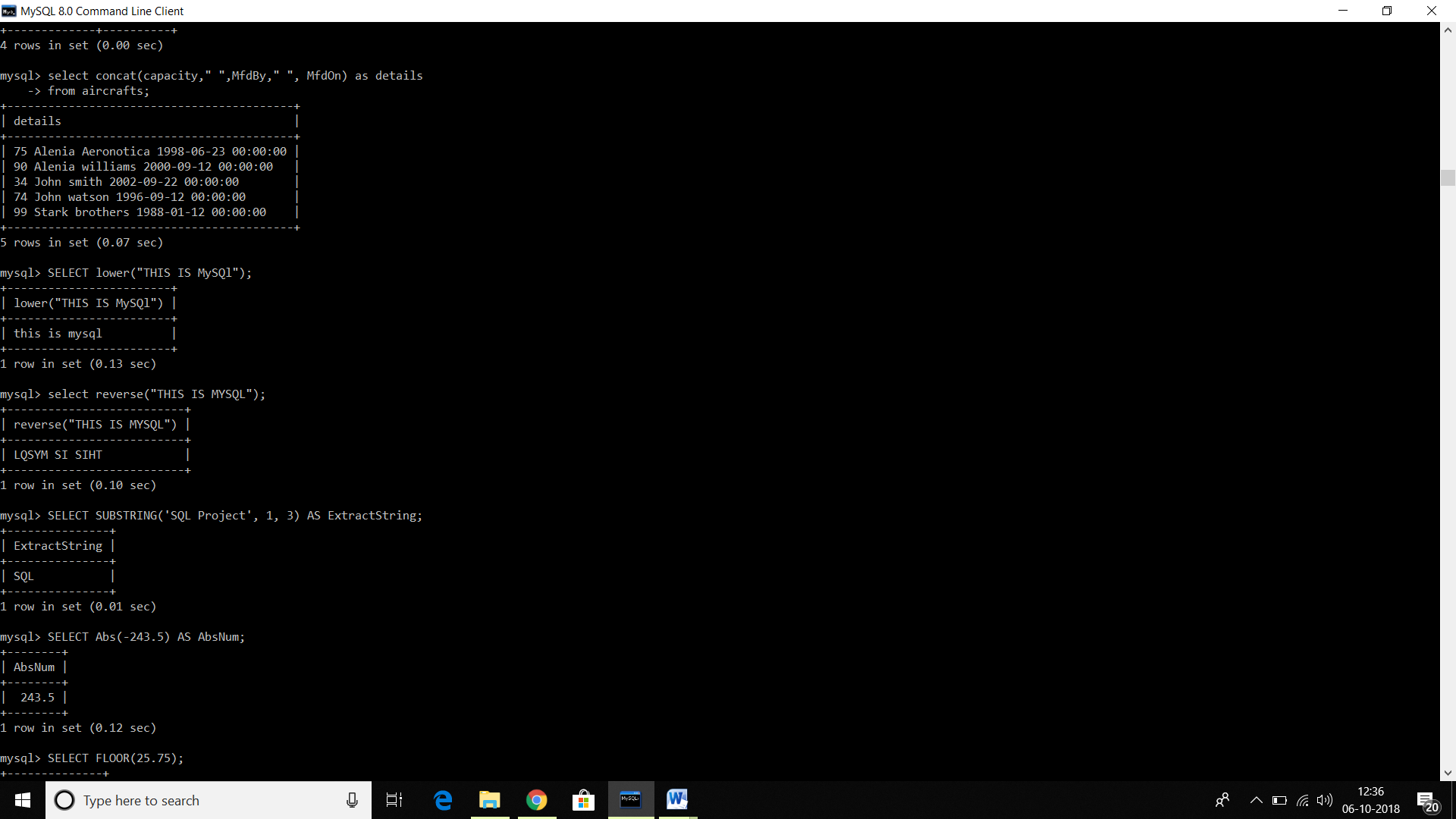
SELECT DATEDIFF(year, '2017/08/25', '2011/08/25') AS DateDiff;

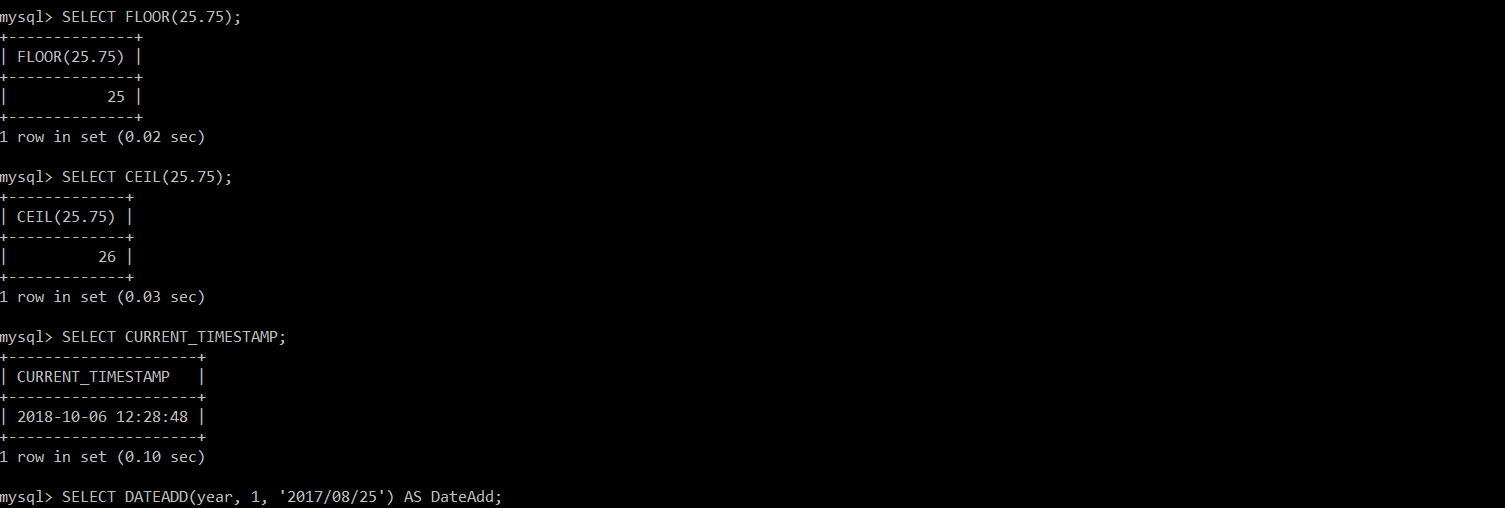
**10:GETDATE:**

The GETDATE() function returns the current database system date and time, in a 'YYYY-MM-DD hh:mm:ss.mmm' format.

GETDATE()

SELECT GETDATE();

****

****

### VIEWS

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

CREATE VIEW Syntax

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

## SQL Updating a View

A view can be updated with the CREATE OR REPLACE VIEW command.

### SQL CREATE OR REPLACE VIEW Syntax

CREATE OR REPLACE VIEW view\_name AS  
SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

#### a) CREATING VIEWS

SQL> create view Bangalore as

1. select Train\_no , Train\_,name
2. from Train
3. where Destination = 'Bangalore'; View created.

#### b) MODIFYING VIEWS

SQL> create or replace view Bangalore as

1. select Train\_no , Train\_name , Source
2. from Train
3. where Destination = 'Bangalore'; View created.

#### c) RETRIVING DATA FROM VIEW

SQL> select \* from IT;

TRAIN\_NO TRAIN\_NAME SOURCE

----------------- ---------------- ----------------

003 Chennai Express Chennai

#### d) REMOVING A VIEW

SQL> drop view Bangalore ; View dropped.

**NORMALIZATION :**

Normalization is a database design technique which organizes tables in a manner that reduces redundancy and dependency of data.

It divides larger tables to smaller tables and links them using relationships.

## **1NF (First Normal Form) Rules**

* Each table cell should contain a single value.
* Each record needs to be unique.

## 2NF (Second Normal Form) Rules

* Rule 1- Be in 1NF
* Rule 2- Single Column Primary Key

## NF (Third Normal Form) Rules

* Rule 1- Be in 2NF
* Rule 2- Has no transitive functional dependencies

### 4NF (Fourth Normal Form) Rules

If no database table instance contains two or more, independent and multivalued data describing the relevant entity, then it is in 4th Normal Form.

### 5NF (Fifth Normal Form) Rules

A table is in 5th Normal Form only if it is in 4NF and it cannot be decomposed into any number of smaller tables without loss of data.

### 6NF (Sixth Normal Form) Proposed

6th Normal Form is not standardized, yet however, it is being discussed by database experts for some time. Hopefully, we would have a clear & standardized definition for 6th Normal Form in the near future.

**APPLICATION DEVELOPMENT :**

**JDBC FRONT-END TECHNOLOGY:**

**Java Database Connectivity** (**JDBC**) is an [application programming interface](https://en.wikipedia.org/wiki/Application_programming_interface) (API) for the programming language [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), which defines how a client may access a [database](https://en.wikipedia.org/wiki/Database). It is a Java-based data access technology used for Java database connectivity. It is part of the [Java Standard Edition](https://en.wikipedia.org/wiki/Java_Standard_Edition) platform, from [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation). It provides methods to query and update data in a database, and is oriented towards [relational databases](https://en.wikipedia.org/wiki/Relational_database). A JDBC-to-[ODBC](https://en.wikipedia.org/wiki/ODBC) bridge enables connections to any ODBC-accessible data source in the [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine) (JVM) host environment

JDBC allows multiple implementations to exist and be used by the same application. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections.

JDBC connections support creating and executing statements. These may be update statements such as SQL's CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT. Additionally, stored procedures may be invoked through a JDBC connection. JDBC represents statements using one of the following classes:

* [Statement](https://docs.oracle.com/javase/10/docs/api/java/sql/Statement.html) – the statement is sent to the database server each and every time.
* [PreparedStatement](https://docs.oracle.com/javase/10/docs/api/java/sql/PreparedStatement.html) – the statement is cached and then the [execution path](https://en.wikipedia.org/wiki/Query_plan) is pre-determined on the database server allowing it to be executed multiple times in an efficient manner.
* [CallableStatement](https://docs.oracle.com/javase/10/docs/api/java/sql/CallableStatement.html) – used for executing [stored procedures](https://en.wikipedia.org/wiki/Stored_procedures) on the database.

Update statements such as INSERT, UPDATE and DELETE return an update count that indicates how many [rows](https://en.wikipedia.org/wiki/Row_(database)) were affected in the database. These statements do not return any other information.

Query statements return a JDBC row result set. The row result set is used to walk over the [result set](https://en.wikipedia.org/wiki/Result_set). Individual [columns](https://en.wikipedia.org/wiki/Column_(database)) in a row are retrieved either by name or by column number. There may be any number of rows in the result set. The row result set has metadata that describes the names of the columns and their types.

There is an extension to the basic JDBC API in the [javax.sql](https://docs.oracle.com/javase/10/docs/api/javax/sql/package-summary.html).

JDBC connections are often managed via a [connection pool](https://en.wikipedia.org/wiki/Connection_pool) rather than obtained directly from the driver.

**FRONT PAGE :**

import java.awt.BorderLayout;

import java.sql.\*;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.GroupLayout;

import javax.swing.GroupLayout.Alignment;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import java.awt.Font;

import java.awt.Color;

import javax.swing.JTextField;

import javax.swing.JButton;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import javax.swing.LayoutStyle.ComponentPlacement;

import javax.swing.JComboBox;

public class ISS extends JFrame {

static ISS frame;

private JPanel contentPane;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

frame = new ISS();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public ISS() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 450, 404);

contentPane = new JPanel();

contentPane.setBackground(Color.LIGHT\_GRAY);

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

JButton btnNewButton = new JButton("LOGIN");

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

secondpage second= new secondpage();

second.setVisible(true);

}

});

JButton btnUserLogin = new JButton("SIGNUP");

btnUserLogin.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

user u= new user();

u.setVisible(true);

}

});

});

JLabel rrRailwayReservationSystem = new JLabel("RAILWAY RESERVATION SYSTEM");

rrRailwayReservationSystem.setFont(new Font("Tahoma", Font.PLAIN, 18));

GroupLayout gl\_contentPane = new GroupLayout(contentPane);

gl\_contentPane.setHorizontalGroup(

gl\_contentPane.createParallelGroup(Alignment.TRAILING)

.addGroup(gl\_contentPane.createSequentialGroup()

.addContainerGap(102, Short.MAX\_VALUE)

.addComponent(lblLibraryMangementSystem, GroupLayout.PREFERRED\_SIZE, 258, GroupLayout.PREFERRED\_SIZE)

.addGap(62))

.addGroup(Alignment.LEADING, gl\_contentPane.createSequentialGroup()

.addGap(60)

.addGroup(gl\_contentPane.createParallelGroup(Alignment.LEADING)

.addComponent(btnUserSignup)

.addComponent(btnUserLogin)

.addComponent(btnNewButton))

.addContainerGap(251, Short.MAX\_VALUE))

);

gl\_contentPane.setVerticalGroup(

gl\_contentPane.createParallelGroup(Alignment.TRAILING)

.addGroup(Alignment.LEADING, gl\_contentPane.createSequentialGroup()

.addGap(22)

.addComponent rrRailwayReservationSystem,

GroupLayout.PREFERRED\_SIZE, 38, GroupLayout.PREFERRED\_SIZE)

.addGap(18)

.addComponent(btnNewButton)

.addGap(40)

.addComponent(btnUserLogin)

.addGap(41)

.addComponent(btnUserSignup)

.addContainerGap(113, Short.MAX\_VALUE))

);

contentPane.setLayout(gl\_contentPane);

}

}

**SIGNUP PAGE:**

**import** javax.swing.\*;

**import** java.awt.event.\*;

**import** java.awt.\*;

**import** java.sql.\*;

**public** **class** Signup **implements** ActionListener {

JFrame frame;

JLabel nameLabel=**new** JLabel("NAME");

JLabel passwordLabel=**new** JLabel("PASSWORD");

JLabel confirmPasswordLabel=**new** JLabel("CONFIRM PASSWORD");

JTextField nameTextField=**new** JTextField();

JPasswordField passwordField=**new** JPasswordField();

JPasswordField confirmPasswordField=**new** JPasswordField();

JButton registerButton=**new** JButton("REGISTER");

JButton resetButton=**new** JButton("RESET");

**private** **final** JLabel lblNewLabel = **new** JLabel("USER ID");

**private** **final** JTextField textField = **new** JTextField();

Signup()

{

createWindow();

setLocationAndSize();

addComponentsToFrame();

actionEvent();

}

**public** **void** createWindow()

{

textField.setBounds(180, 80, 165, 22);

textField.setColumns(10);

frame=**new** JFrame();

frame.setTitle("Registration Form");

frame.setBounds(40,40,380,600);

frame.getContentPane().setBackground(Color.***pink***);

frame.getContentPane().setLayout(**null**);

frame.setVisible(**true**);

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

frame.setResizable(**false**);

}

**public** **void** setLocationAndSize()

{

nameLabel.setBounds(12,19,40,70);

passwordLabel.setBounds(12,91,100,70);

confirmPasswordLabel.setBounds(12,140,140,70);

nameTextField.setBounds(180,43,165,23);

passwordField.setBounds(180,115,165,23);

confirmPasswordField.setBounds(180,164,165,23);

registerButton.setBounds(20,365,100,35);

resetButton.setBounds(212,365,100,35);

}

**public** **void** addComponentsToFrame()

{

frame.getContentPane().add(nameLabel);

frame.getContentPane().add(passwordLabel);

frame.getContentPane().add(confirmPasswordLabel);

frame.getContentPane().add(nameTextField);

frame.getContentPane().add(passwordField);

frame.getContentPane().add(confirmPasswordField);

frame.getContentPane().add(registerButton);

frame.getContentPane().add(resetButton);

lblNewLabel.setBounds(12, 84, 56, 16);

frame.getContentPane().add(lblNewLabel);

frame.getContentPane().add(textField);

}

**public** **void** actionEvent()

{

registerButton.addActionListener(**this**);

resetButton.addActionListener(**this**);

}

@Override

**public** **void** actionPerformed(ActionEvent e) {

Connection connection=**null**;

**if**(e.getSource()==registerButton)

{

**try** {

String u="root";

String p="xxx";

connection=DriverManager.*getConnection*("jdbc:mysql://localhost:3306/airbus330","u","p");

PreparedStatement Pstatement=connection.prepareStatement("insert into d values(?,?,?,?)");

Pstatement.setString(1,nameTextField.getText());

Pstatement.setString(2,textField.getText() );

Pstatement.setString(3,passwordField.~~getText~~());

Pstatement.setString(4,confirmPasswordField.~~getText~~());

//Checking for the Password match

**if**(passwordField.~~getText~~().equalsIgnoreCase(confirmPasswordField.~~getText~~()))

{

//Executing query

Pstatement.executeUpdate();

JOptionPane.*showMessageDialog*(**null**,"Data Registered Successfully");

}

**else**

{

JOptionPane.*showMessageDialog*(**null**,"password did not match");

}

} **catch** (SQLException e1) {

e1.printStackTrace();

}

}

**if**(e.getSource()==resetButton)

{

//Clearing Fields

nameTextField.setText("");

textField.setText("");

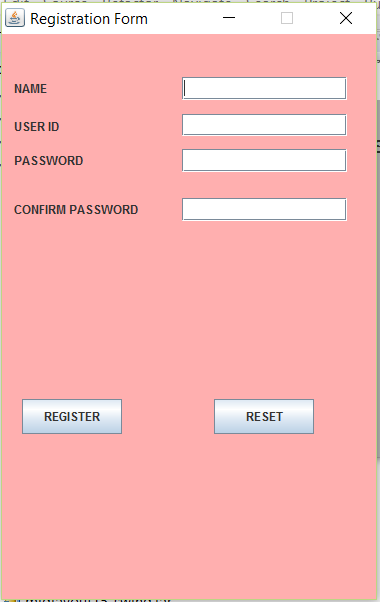
passwordField.setText("");

confirmPasswordField.setText("");

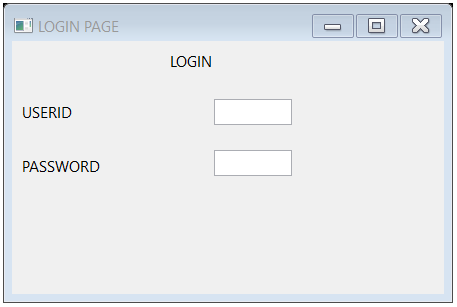
}

}

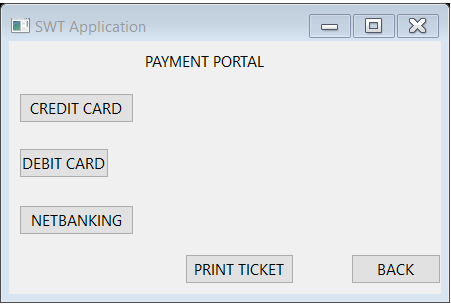
}



**LOGIN PAGE:**

****

**PAYMENT PAGE:**

****

**DATABASE CONNECTIVITY:**

**public** **void** actionPerformed(ActionEvent e) {

Connection connection=**null**;

**if**(e.getSource()==registerButton)

{

**try** {

String u="root";

String p="xxx";

connection=DriverManager.*getConnection*("jdbc:mysql://localhost:3306/railway330","u","p");

PreparedStatement Pstatement=connection.prepareStatement("insert into d values(?,?,?,?)");

Pstatement.setString(1,nameTextField.getText());

Pstatement.setString(2,textField.getText() );

Pstatement.setString(3,passwordField.~~getText~~());

Pstatement.setString(4,confirmPasswordField.~~getText~~());

//Checking for the Password match

**if**(passwordField.~~getText~~().equalsIgnoreCase(confirmPasswordField.~~getText~~()))

{

//Executing query

Pstatement.executeUpdate();

JOptionPane.*showMessageDialog*(**null**,"Data Registered Successfully");

}

**else**

{

JOptionPane.*showMessageDialog*(**null**,"password did not match");

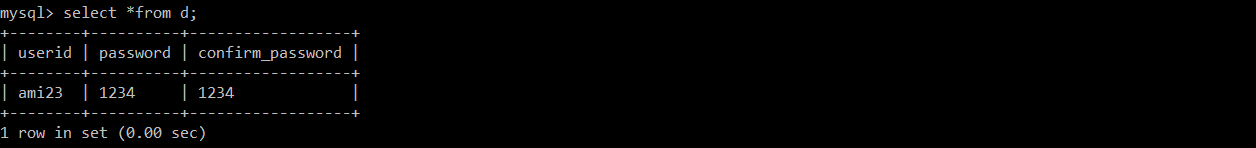
}

} **catch** (SQLException e1) {

e1.printStackTrace();

}

}

****

**CONCLUSION:**

My Railway Management system project has been one of the rewarding experiences I have had. I have learnt a lot of things in the entire course of my Project. My desire to learn one of the fast-evolving technologies like JDBC and JAVA SWINGS has been satisfied. I have learnt the various coding techniques in java swings.

I have experienced a situation similar to the real-time work environment, where programmers and developers work under pressure and a specified deadline. Being new to the technology I had some difficulties with the coding part initially. Going through the entire life-cycle of the software development has given me a lot of knowledge and experience which will be useful for my future. I have also improved my coding skills through this project.

The Railway reservation system project has also helped me realize that documentation for a project is as equally important as the coding of the project. Testing the Railway Reservation System project has helped me gain a lot of knowledge about the stress and load testing of the web applications. I was able to figure out the procedure with some research online. Overall, it was a great experience and I learned a lot from this project.