**PROJECT 2**

**ADVANCE DATABASE**

**GROUP 5**

Vaishali Gupta (00001588183)

Sarvesh Kulkarni (00001588389)

Siddhant Kshatriya (00001588464)

Thomas Francis (00001561030)

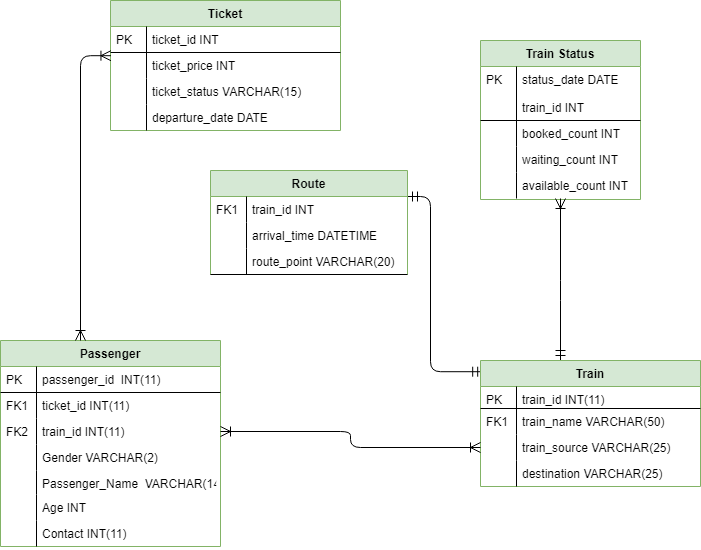
Manas Sandhawani (00001588394)

**Abstract**

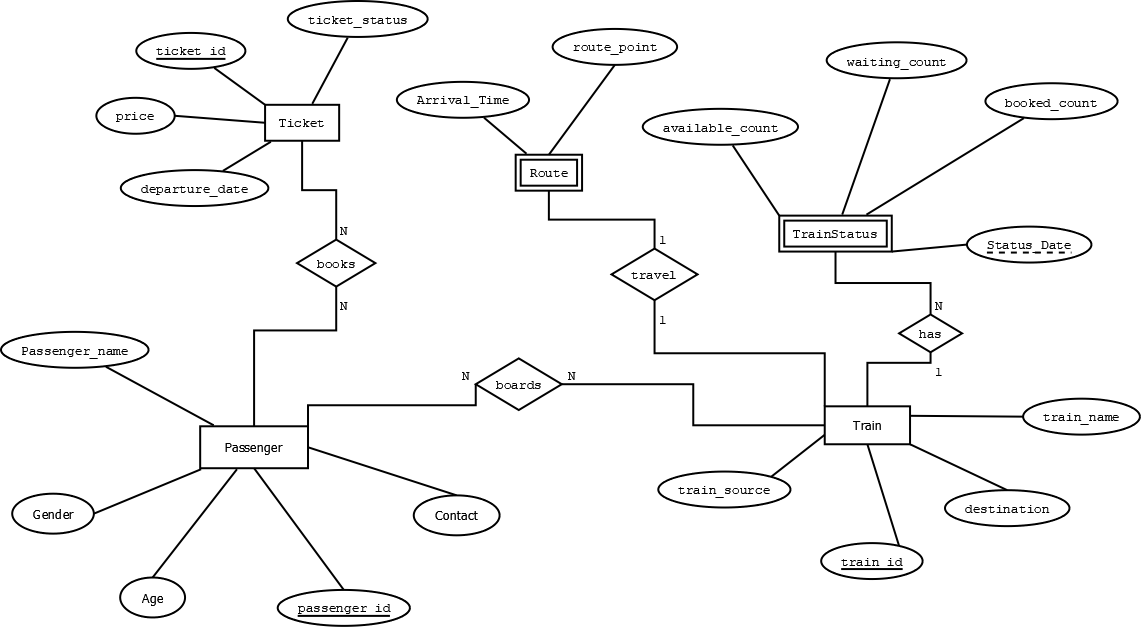
The project focuses on Train Management System which will allow the track of all train schedules, their route and passenger details. This will also allow the Train Administrator to manage, edit and add routes. Also, we can add new passenger and edit ticket details. This System simplifies the train management by maintaining all the records.

The database used in this project is MySQL and JDBC driver used is *mysql-connector-java-8.0.19.* This project uses 5 different tables. DDL (Data Definition Language), DML (Data Manipulation Language) are queried on database.

**Schema Diagram**



**ER Diagram:**



**JDBC Source Code for Creating Tables and Insert Queries**

***JDBCConnection.java***

**package** main.java;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**public** **class** JDBCConnection {

**private** Connection connect = **null**;

**private** Statement statement = **null**;

**private** ResultSet resultSet = **null**;

**private** **final** String connectionURL = "jdbc:mysql://localhost:3306/project2?user=root&password=siddhant16";

**public** **void** readDataBase() **throws** Exception {

**try** {

// This will load the MySQL driver, each DB has its own driver

Class.*forName*("com.mysql.jdbc.Driver");

// Setup the connection with the DB

connect = DriverManager.*getConnection*(connectionURL);

statement = connect.createStatement();

DBUtil.*fireCreateSQL*(connect);

System.***out***.println("############################### Create Queries Run ##########################################");

printStateOfDBTables();

DBUtil.*fireInsertQueries*(connect);

System.***out***.println("############################### Insert Queries Run ##########################################");

printStateOfDBTables();

System.***out***.println("############################### Running Custom Queries##########################################");

DBUtil.*fireCustomQueries*(connect);

} **catch** (Exception e) {

**throw** e;

} **finally** {

close();

}

}

**private** **void** printStateOfDBTables() **throws** SQLException {

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println("Select \* from Train");

resultSet = statement.executeQuery("select \* from Train");

printTrainData(resultSet);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println("Select \* from passenger");

resultSet = statement.executeQuery("select \* from passenger");

printPassengerData(resultSet);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println("Select \* from route");

resultSet = statement.executeQuery("select \* from route");

printRouteData(resultSet);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println("Select \* from ticket");

resultSet = statement.executeQuery("select \* from ticket");

printTicketData(resultSet);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println("Select \* from trainstatus");

resultSet = statement.executeQuery("select \* from trainstatus");

printTicketStatus(resultSet);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

**private** **void** printTrainData(ResultSet resultSet) **throws** SQLException {

**while** (resultSet.next()) {

System.***out***.println("TrainId: " + resultSet.getInt("train\_id") + " Train Name: " + resultSet.getString("train\_name") + " Source: " + resultSet.getString("train\_source") + " Destination: " + resultSet.getString("destination"));

}

}

**private** **void** printPassengerData(ResultSet resultSet) **throws** SQLException {

**while** (resultSet.next()) {

System.***out***.println("Passenger Id: " + resultSet.getInt("passenger\_id") + " Train Id: " + resultSet.getInt("train\_id") + " ticket id: " + resultSet.getInt("ticket\_id") + " Passenger: " + resultSet.getString("Passenger\_name") + " Gender : " + resultSet.getString("Gender")+ " Age : " + resultSet.getInt("Age")+ " Contact: " + resultSet.getInt("Contact"));

}

}

**private** **void** printRouteData(ResultSet resultSet) **throws** SQLException {

**while** (resultSet.next()) {

System.***out***.println("Train Id: " + resultSet.getInt("train\_id") + " Arrival time: " + resultSet.getDate("Arrival\_Time") + " Route point: " + resultSet.getString("route\_point"));

}

}

**private** **void** printTicketData(ResultSet resultSet) **throws** SQLException {

**while** (resultSet.next()) {

System.***out***.println("Ticket Id: " + resultSet.getInt("ticket\_id") + " ticket\_price: " + resultSet.getInt("ticket\_price") + " ticket status: " + resultSet.getString("ticket\_status") + " ticket status: " + resultSet.getDate("deaparture\_date"));

}

}

**private** **void** printTicketStatus(ResultSet resultSet) **throws** SQLException {

**while** (resultSet.next()) {

System.***out***.println("Train Id: " + resultSet.getInt("train\_id") + " Available count: " + resultSet.getInt("avaible\_count") + " Booked count: " + resultSet.getInt("booked\_count") + " Waiting\_Count "+ resultSet.getInt("waiting\_count") + " Status Date: " + resultSet.getDate("Status\_Date"));

}

}

// You need to close the resultSet

**private** **void** close() {

**try** {

**if** (resultSet != **null**) {

resultSet.close();

}

**if** (statement != **null**) {

statement.close();

}

**if** (connect != **null**) {

connect.close();

}

} **catch** (Exception e) {

}

}

**public** **static** **void** main(String[] args) **throws** Exception {

JDBCConnection connection = **new** JDBCConnection();

connection.readDataBase();

}

}

***DBUtil.java***

**package** main.java;

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.FileReader;

**import** java.sql.SQLException;

**import** java.sql.Connection;

**import** java.sql.Statement;

**public** **class** DBUtil {

**public** **static** **void** selectTrainQuery() {

}

**public** **static** **void** fireInsertQueries(Connection connection) {

String s = **new** String();

StringBuffer sb = **new** StringBuffer();

**try**

{

FileReader fr = **new** FileReader(**new** File("D:\\College Data\\Masters\\Quarter 2\\ADB\\Projects\\P2\\P2CreateTableQueries.sql"));

BufferedReader br = **new** BufferedReader(fr);

**while**((s = br.readLine()) != **null**)

{

sb.append(s);

}

br.close();

// here is our splitter ! We use ";" as a delimiter for each request

// then we are sure to have well formed statements

String[] inst = sb.toString().split(";");

Statement st = connection.createStatement();

**for**(**int** i = 0; i < inst.length; i++)

{

// we ensure that there is no spaces before or after the request string

// in order to not execute empty statements

**if**(!inst[i].trim().equals(""))

{

st.executeUpdate(inst[i]);

System.***out***.println(">>"+inst[i]);

}

}

}

**catch**(Exception e)

{

System.***out***.println("\*\*\* Error : "+e.toString());

System.***out***.println("\*\*\* ");

System.***out***.println("\*\*\* Error : ");

e.printStackTrace();

System.***out***.println("################################################");

System.***out***.println(sb.toString());

}

}

**public** **static** **void** fireCustomQueries(Connection connection) {

String s = **new** String();

StringBuffer sb = **new** StringBuffer();

**try**

{

FileReader fr = **new** FileReader(**new** File("D:\\College Data\\Masters\\Quarter 2\\ADB\\Projects\\P2\\P2CreateTableQueries.sql"));

BufferedReader br = **new** BufferedReader(fr);

**while**((s = br.readLine()) != **null**)

{

sb.append(s);

}

br.close();

// here is our splitter ! We use ";" as a delimiter for each request

// then we are sure to have well formed statements

String[] inst = sb.toString().split(";");

Statement st = connection.createStatement();

**for**(**int** i = 0; i < inst.length; i++)

{

// we ensure that there is no spaces before or after the request string

// in order to not execute empty statements

**if**(!inst[i].trim().equals(""))

{

st.executeUpdate(inst[i]);

System.***out***.println(">>"+inst[i]);

}

}

}

**catch**(Exception e)

{

System.***out***.println("\*\*\* Error : "+e.toString());

System.***out***.println("\*\*\* ");

System.***out***.println("\*\*\* Error : ");

e.printStackTrace();

System.***out***.println("################################################");

System.***out***.println(sb.toString());

}

}

**public** **static** **void** fireCreateSQL(Connection connection) **throws** SQLException

{

String s = **new** String();

StringBuffer sb = **new** StringBuffer();

**try**

{

FileReader fr = **new** FileReader(**new** File("C:\\Users\\Thoma\\Downloads\\ADB\_Project2-20200204T032141Z-001\\ADB\_Project2\\src\\main\\resources\\Create.sql"));

BufferedReader br = **new** BufferedReader(fr);

**while**((s = br.readLine()) != **null**)

{

sb.append(s);

}

br.close();

// here is our splitter ! We use ";" as a delimiter for each request

// then we are sure to have well formed statements

String[] inst = sb.toString().split(";");

Statement st = connection.createStatement();

**for**(**int** i = 0; i < inst.length; i++)

{

// we ensure that there is no spaces before or after the request string

// in order to not execute empty statements

**if**(!inst[i].trim().equals(""))

{

st.executeUpdate(inst[i]);

System.***out***.println(">>"+inst[i]);

}

}

}

**catch**(Exception e)

{

System.***out***.println("\*\*\* Error : "+e.toString());

System.***out***.println("\*\*\* ");

System.***out***.println("\*\*\* Error : ");

e.printStackTrace();

System.***out***.println("################################################");

System.***out***.println(sb.toString());

}

}

}

**CREATE TABLE STATEMENTS**

Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is automatically registered via the SPI and manual loading of the driver class is generally unnecessary.

############################### Create Queries Run ##########################################

>>CREATE TABLE Train

(train\_id INT NOT NULL,

train\_name VARCHAR(50) NOT NULL,

train\_source VARCHAR(25) NOT NULL,

destination VARCHAR(25) NOT NULL ,

PRIMARY KEY(train\_id))

>>CREATE TABLE TrainStatus

(train\_id INT NOT NULL,

available\_count INT,

booked\_count INT,

waiting\_count INT,

Status\_Date DATE NOT NULL,

FOREIGN KEY(train\_id) REFERENCES Train(train\_id),

PRIMARY KEY(train\_id, Status\_Date))

>>CREATE TABLE Route

(train\_id INT NOT NULL,

Arrival\_Time DATETIME,

route\_point VARCHAR(20) NOT NULL,

FOREIGN KEY(train\_id) REFERENCES Train(train\_id))

>>CREATE TABLE Ticket

(ticket\_id INT NOT NULL,

ticket\_price INT NOT NULL,

ticket\_status VARCHAR(15),

departure\_date DATE,

PRIMARY KEY(ticket\_id))

>>CREATE TABLE Passenger

(passenger\_id INT NOT NULL,

train\_id INT NOT NULL,

ticket\_id INT NOT NULL,

Passenger\_name VARCHAR(14) NOT NULL,

Gender VARCHAR(2),

Age INT,Contact INT,

PRIMARY KEY(passenger\_id),

FOREIGN KEY(train\_id) REFERENCES train(train\_id),

FOREIGN KEY(ticket\_id) REFERENCES Ticket(ticket\_id))

############################### Insert Queries Run ##########################################

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Select \* from Train

TrainId: 91101 Train Name: Santa Clara Express Source: Santa Clara Destination: San Franscisco

TrainId: 91208 Train Name: Southern Cali Express Source: Los Angeles Destination: San Diego

TrainId: 92307 Train Name: San Jose Amtrak Source: San Jose Destination: Fremont

TrainId: 93461 Train Name: Oakland Santa Clara Express Source: Oakland Destination: Santa Clara

TrainId: 96782 Train Name: California Special Source: Sacramento Destination: San Diego

TrainId: 97091 Train Name: Santa Crus ACE Source: Santa Cruz Destination: San Franscisco

TrainId: 99093 Train Name: Caltrain Weekends Source: San Franscisco Destination: San Jose

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Select \* from passenger

Passenger Id: 10011 Train Id: 91101 ticket id: 30010 Passenger: Ross Gellar Gender : M Age : 45 Contact: 408777626

Passenger Id: 10292 Train Id: 99093 ticket id: 30110 Passenger: Daniel Gender : M Age : 65 Contact: 557728194

Passenger Id: 11352 Train Id: 91101 ticket id: 30567 Passenger: Angelina Gender : F Age : 28 Contact: 777639271

Passenger Id: 11737 Train Id: 99093 ticket id: 33921 Passenger: Virat Kohli Gender : M Age : 40 Contact: 136288456

Passenger Id: 12113 Train Id: 91208 ticket id: 31244 Passenger: Tony Stark Gender : M Age : 47 Contact: 467282517

Passenger Id: 12143 Train Id: 91101 ticket id: 30157 Passenger: Gunther Gender : M Age : 41 Contact: 779667543

Passenger Id: 12312 Train Id: 92307 ticket id: 30215 Passenger: Carmo Gender : F Age : 48 Contact: 552637882

Passenger Id: 12326 Train Id: 92307 ticket id: 30372 Passenger: Bendict Gender : M Age : 18 Contact: 667887242

Passenger Id: 12341 Train Id: 96782 ticket id: 30211 Passenger: Lisa Haidan Gender : F Age : 25 Contact: 669262913

Passenger Id: 12381 Train Id: 96782 ticket id: 31382 Passenger: Susan Gender : F Age : 11 Contact: 123435493

Passenger Id: 17222 Train Id: 96782 ticket id: 30128 Passenger: Scarlett Gender : F Age : 23 Contact: 618361832

Passenger Id: 18832 Train Id: 91101 ticket id: 30345 Passenger: David Gender : M Age : 26 Contact: 726193616

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Select \* from route

Train Id: 91101 Arrival time: 2020-01-01 Route point: Santa Clara

Train Id: 91101 Arrival time: 2020-01-01 Route point: Palo Alto

Train Id: 91101 Arrival time: 2020-01-01 Route point: Redwood City

Train Id: 91101 Arrival time: 2020-01-01 Route point: San Fransciso

Train Id: 91101 Arrival time: 2019-12-13 Route point: Santa Clara

Train Id: 91101 Arrival time: 2019-12-13 Route point: Palo Alto

Train Id: 91101 Arrival time: 2019-12-13 Route point: Redwood City

Train Id: 91101 Arrival time: 2019-12-13 Route point: San Fransciso

Train Id: 92307 Arrival time: 2019-01-11 Route point: San Jose

Train Id: 92307 Arrival time: 2019-01-11 Route point: North San Jose

Train Id: 92307 Arrival time: 2019-01-11 Route point: Milpitas

Train Id: 92307 Arrival time: 2020-01-11 Route point: Fremont

Train Id: 97091 Arrival time: 2020-12-24 Route point: Santa Cruz

Train Id: 97091 Arrival time: 2020-12-24 Route point: Los Gatos

Train Id: 97091 Arrival time: 2020-12-24 Route point: SunnyVale

Train Id: 97091 Arrival time: 2020-12-24 Route point: Mountain View

Train Id: 97091 Arrival time: 2020-01-01 Route point: Palt Alto

Train Id: 97091 Arrival time: 2020-01-01 Route point: San Mateo

Train Id: 97091 Arrival time: 2020-01-01 Route point: Milbrae

Train Id: 97091 Arrival time: 2020-01-01 Route point: San Franscisco

Train Id: 93461 Arrival time: 2020-03-31 Route point: Oakland

Train Id: 93461 Arrival time: 2020-03-31 Route point: Hayward

Train Id: 93461 Arrival time: 2020-03-31 Route point: Fremont

Train Id: 93461 Arrival time: 2020-03-31 Route point: Milpitas

Train Id: 93461 Arrival time: 2020-03-31 Route point: San Jose

Train Id: 93461 Arrival time: 2020-03-31 Route point: Santa Clara

Train Id: 93461 Arrival time: 2020-03-31 Route point: San Jose

Train Id: 99093 Arrival time: 2020-01-01 Route point: San Franscisco

Train Id: 99093 Arrival time: 2020-01-01 Route point: Redwood City

Train Id: 99093 Arrival time: 2020-01-01 Route point: Palo Alto

Train Id: 99093 Arrival time: 2020-01-01 Route point: San Jose

Train Id: 96782 Arrival time: 2019-12-28 Route point: Sacramento

Train Id: 96782 Arrival time: 2019-12-28 Route point: Fremont

Train Id: 96782 Arrival time: 2019-12-28 Route point: San Jose

Train Id: 96782 Arrival time: 2019-12-28 Route point: Bakersfield

Train Id: 96782 Arrival time: 2019-12-28 Route point: Los Angeles

Train Id: 96782 Arrival time: 2019-12-28 Route point: San Diego

Train Id: 91208 Arrival time: 2020-01-01 Route point: Los Angeles

Train Id: 91208 Arrival time: 2020-01-01 Route point: Riverside

Train Id: 91208 Arrival time: 2020-01-01 Route point: San Clemente

Train Id: 91208 Arrival time: 2020-01-01 Route point: San Diego

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Select \* from ticket

Ticket Id: 30010 ticket\_price: 25 ticket status: Booked ticket status: 2020-01-01

Ticket Id: 30110 ticket\_price: 35 ticket status: Waiting ticket status: 2020-01-01

Ticket Id: 30128 ticket\_price: 5 ticket status: Booked ticket status: 2019-12-13

Ticket Id: 30157 ticket\_price: 45 ticket status: Waiting ticket status: 2020-01-11

Ticket Id: 30211 ticket\_price: 14 ticket status: Cancelled ticket status: 2020-03-31

Ticket Id: 30215 ticket\_price: 24 ticket status: Cancelled ticket status: 2020-01-11

Ticket Id: 30232 ticket\_price: 14 ticket status: Cancelled ticket status: 2020-12-24

Ticket Id: 30244 ticket\_price: 35 ticket status: Waiting ticket status: 2020-12-24

Ticket Id: 30345 ticket\_price: 37 ticket status: Booked ticket status: 2020-03-31

Ticket Id: 30372 ticket\_price: 34 ticket status: Booked ticket status: 2020-01-01

Ticket Id: 30552 ticket\_price: 37 ticket status: Booked ticket status: 2020-02-09

Ticket Id: 30567 ticket\_price: 13 ticket status: Booked ticket status: 2020-03-31

Ticket Id: 30812 ticket\_price: 21 ticket status: Booked ticket status: 2020-12-24

Ticket Id: 31244 ticket\_price: 15 ticket status: Booked ticket status: 2019-12-13

Ticket Id: 31382 ticket\_price: 12 ticket status: Booked ticket status: 2020-02-09

Ticket Id: 32345 ticket\_price: 25 ticket status: Booked ticket status: 2020-01-01

Ticket Id: 33921 ticket\_price: 56 ticket status: Booked ticket status: 2020-01-01

Ticket Id: 39314 ticket\_price: 23 ticket status: Booked ticket status: 2020-01-01

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Select \* from trainstatus

Train Id: 91101 Available count: 15 Booked count: 85 Waiting\_Count 0 Status Date: 2019-12-13

Train Id: 91101 Available count: 25 Booked count: 65 Waiting\_Count 10 Status Date: 2020-01-01

Train Id: 91101 Available count: 0 Booked count: 100 Waiting\_Count 0 Status Date: 2020-12-24

Train Id: 91101 Available count: 13 Booked count: 73 Waiting\_Count 14 Status Date: 2020-12-28

Train Id: 91208 Available count: 34 Booked count: 66 Waiting\_Count 0 Status Date: 2020-01-01

Train Id: 91208 Available count: 48 Booked count: 44 Waiting\_Count 8 Status Date: 2020-01-17

Train Id: 92307 Available count: 34 Booked count: 55 Waiting\_Count 11 Status Date: 2020-01-11

Train Id: 92307 Available count: 6 Booked count: 77 Waiting\_Count 17 Status Date: 2020-01-17

Train Id: 93461 Available count: 45 Booked count: 48 Waiting\_Count 7 Status Date: 2020-01-19

Train Id: 93461 Available count: 18 Booked count: 79 Waiting\_Count 3 Status Date: 2020-02-19

Train Id: 93461 Available count: 20 Booked count: 78 Waiting\_Count 2 Status Date: 2020-03-31

Train Id: 96782 Available count: 3 Booked count: 97 Waiting\_Count 0 Status Date: 2020-01-01

Train Id: 96782 Available count: 27 Booked count: 68 Waiting\_Count 5 Status Date: 2020-02-09

Train Id: 96782 Available count: 89 Booked count: 10 Waiting\_Count 1 Status Date: 2020-12-28

Train Id: 97091 Available count: 1 Booked count: 94 Waiting\_Count 5 Status Date: 2020-01-01

Train Id: 97091 Available count: 25 Booked count: 65 Waiting\_Count 10 Status Date: 2020-12-24

Train Id: 99093 Available count: 3 Booked count: 85 Waiting\_Count 12 Status Date: 2020-01-01

Train Id: 99093 Available count: 11 Booked count: 88 Waiting\_Count 1 Status Date: 2020-01-22

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CUSTOM QUERIES**

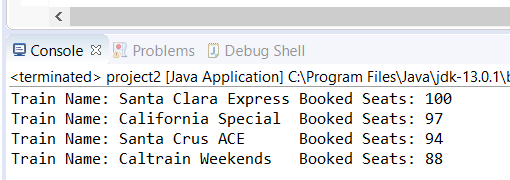
1. **Top 4 trains having maximum booked reservations**

**SELECT** Train\_name, booked\_count **FROM** Train t

**INNER JOIN** TrainStatus ts **ON**

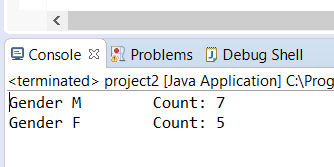
t.train\_id = ts.train\_id

**ORDER BY** Booked\_count **DESC** **limit** 4;



1. **Count number of Male and Female passengers and their Ratio**

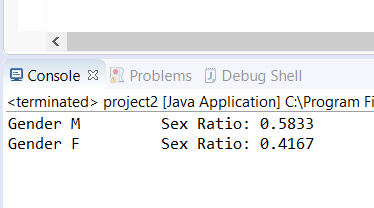
**SELECT** Gender, **COUNT**(passenger\_ID) **FROM** passenger **GROUP BY** gender



1. **Ratio of Male to Female**

SELECT gender, Count(\*) / (select count(\*) FROM Passenger) AS Sex\_Ratio FROM passenger

GROUP BY gender



1. **Display distinct Female Passenger Name and train Details (3 Way Join)**

SELECT DISTINCT(passenger\_name), route\_point, train\_name FROM passenger p

JOIN train t

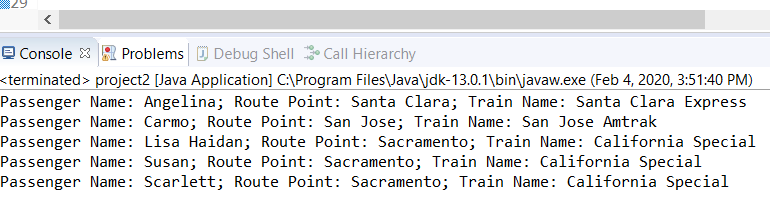
ON t.train\_id = p.train\_id

JOIN route r

ON t.train\_id = r.train\_id

WHERE gender="F"

GROUP BY Passenger\_NAme;



1. **Display Trains who takes more than 4 stops from source to destination**

SELECT train\_name, count(\*) Stops

FROM train T

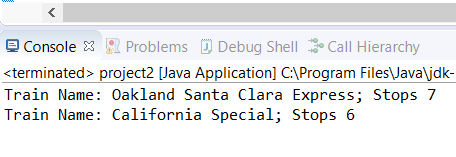
JOIN

route r

ON r.train\_id = t.train\_id

GROUP BY r.train\_id, Date(arrival\_time)

HAVING Stops > 4



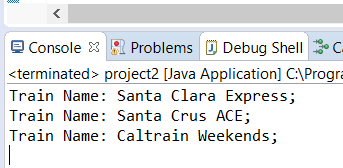
1. **Display Train Names Going to or Coming From San Francisco (Nested Query)**

SELECT train\_name FROM train

WHERE train\_id IN

(SELECT route.train\_id FROM route

WHERE route.route\_point="San Francisco");



1. **Display Passenger and train details combined where ticket price is equal to 35 (4 Way Join)**

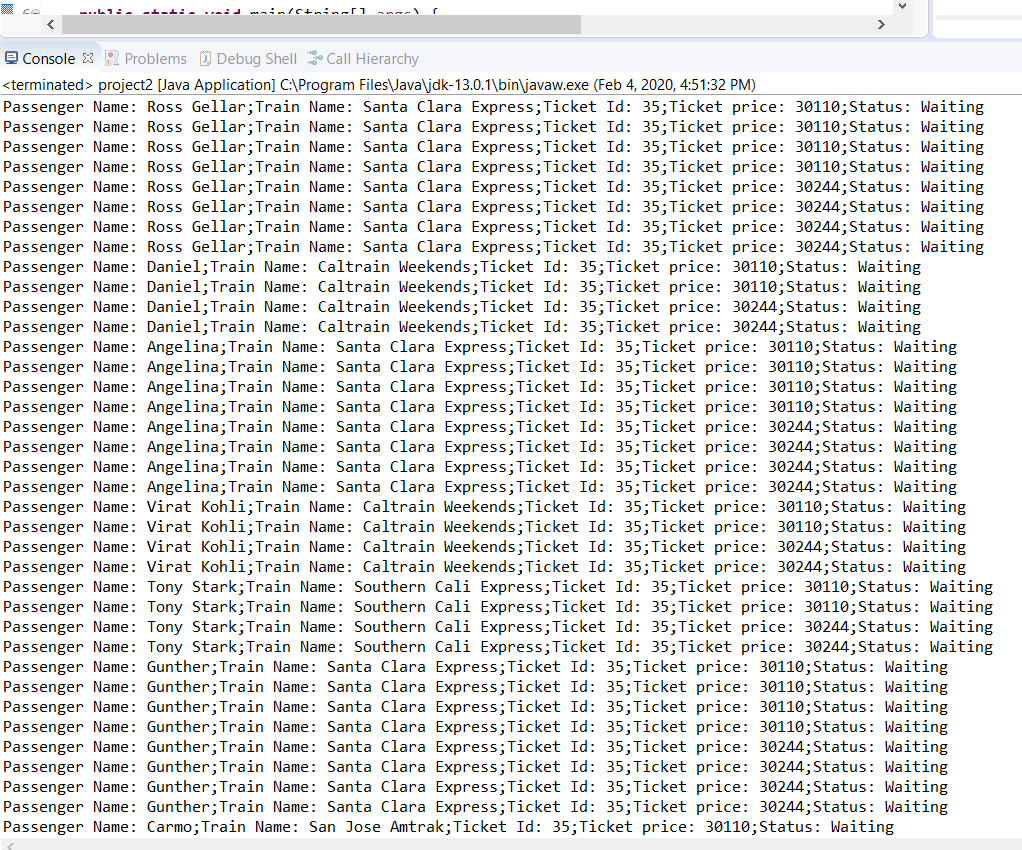
SELECT DISTINCT(Passenger\_name), Train\_name, Ticket\_price, ti.Ticket\_id, Ticket\_status, Status\_date FROM Passenger p

INNER JOIN Train t ON t.train\_id = p.train\_id

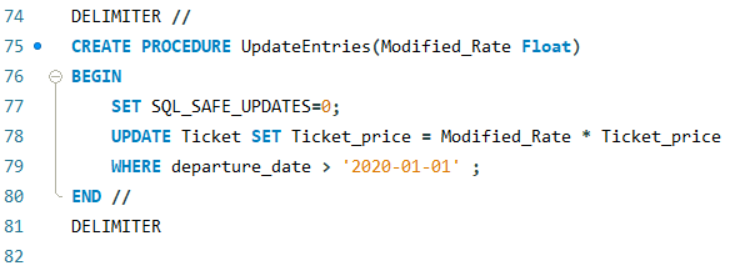
INNER JOIN Ticket ti ON p.ticket\_id = p.ticket\_id

INNER JOIN TrainStatus ts ON t.train\_id = ts.train\_id

WHERE Ticket\_price = 35



1. **Create Procedure to Update the ticket price which is after 1st Jan 2020**



**package** projectJDBC;

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

**public** **class** project2 {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**try** {

Connection conn = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/project2","root","siddhant16");

Statement query = conn.createStatement();

ResultSet res = query.executeQuery

("SELECT \*FROM Ticket ORDER BY ticket\_id");

System.***out***.println("Before Stored Procedure is Called");

**while**(res.next())

{

System.***out***.println("Ticket Id: "+ res.getString(1) +";"+

"Ticket Price: "+ res.getString(2) +";"+

"Ticket Status: " +res.getString(3) + ";"+

"Depature Date: "+ res.getString(4) +";");

}

String quer = "{ call UpdateEntries(?) }";

CallableStatement stmt = conn.prepareCall(quer);

stmt.setDouble(1, 1.5);

stmt.execute();

res = query.executeQuery

("SELECT \*FROM Ticket ORDER BY ticket\_id");

System.***out***.println("After Stored Procedure is executed");

**while**(res.next())

{

System.***out***.println("Ticket Id: "+ res.getString(1) +" ;"+

"Ticket Price: "+ res.getString(2) +" ;"+

"Ticket Status: " +res.getString(3) + " ;"+

"Depature Date: "+ res.getString(4) +" ;");

}

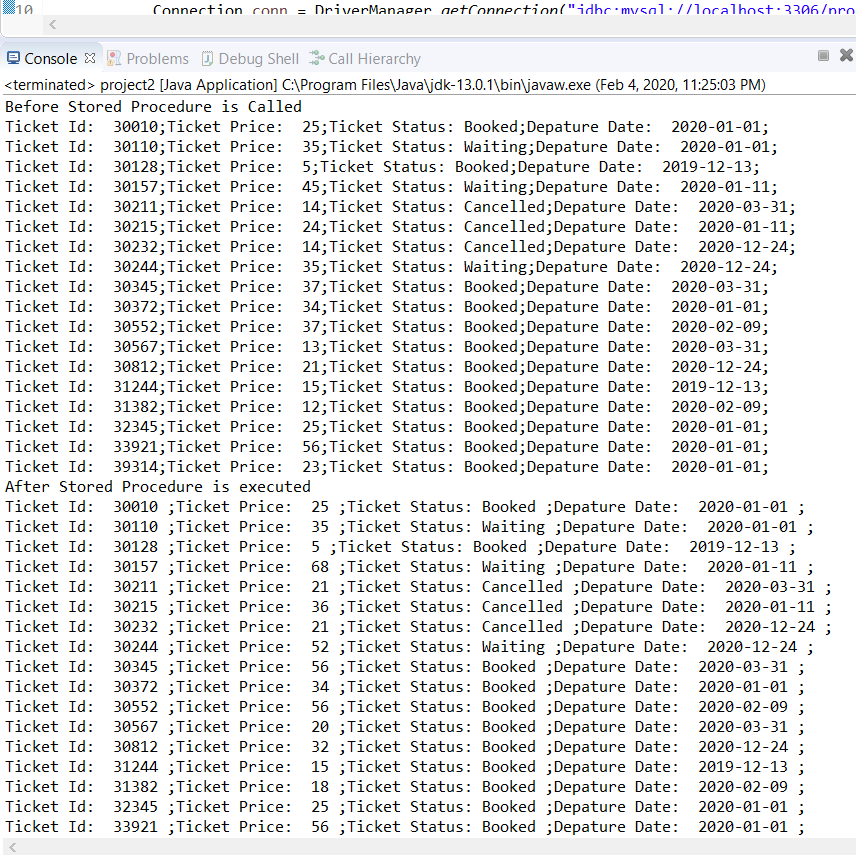
}

**catch**(Exception e) {

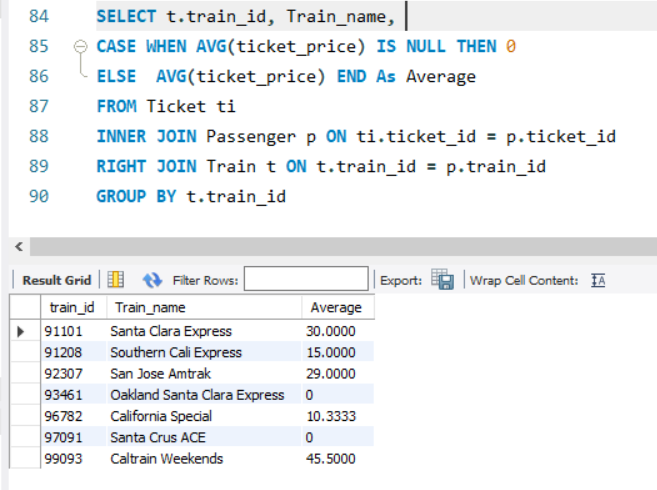
e.printStackTrace();

}

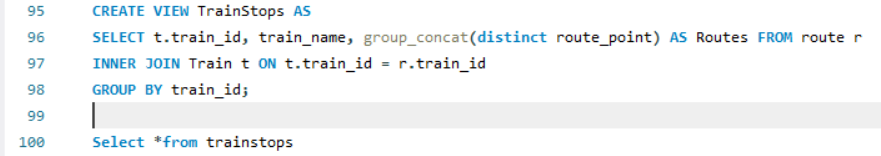
}

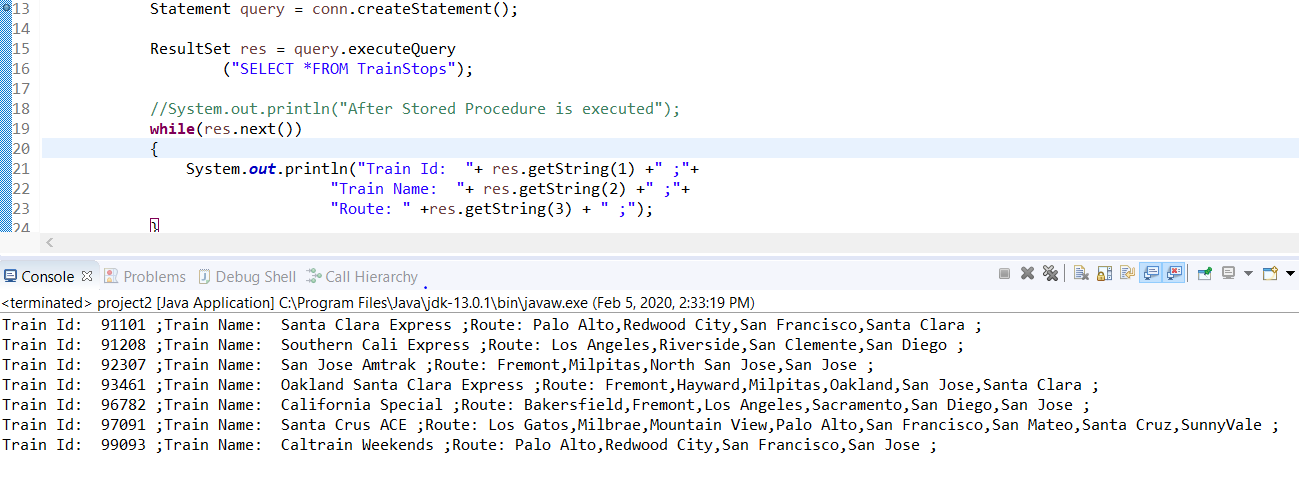
}

1. **Display Average Ticket Price for all the trains, if no tickets are present for train show as 0**

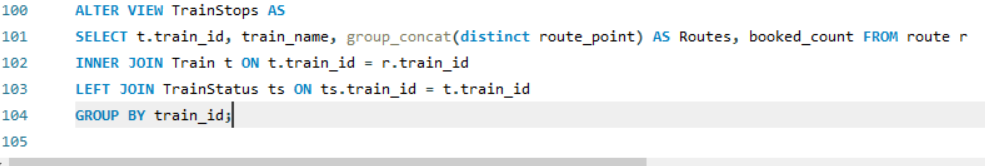


1. **Create View of Train Name and their Routes.**

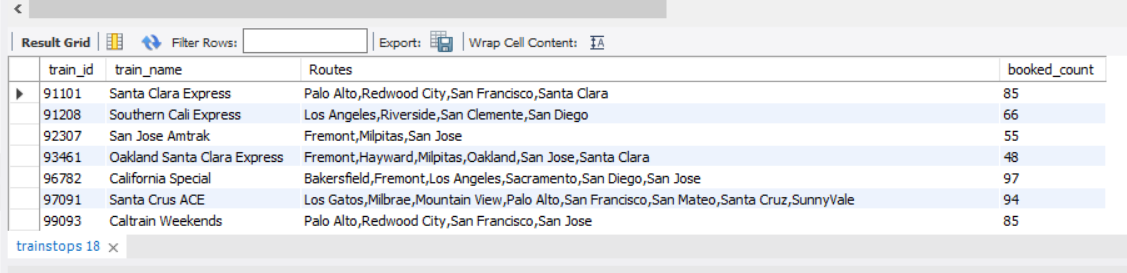




1. **Update View**



SELECT \*FROM TrainStops



1. **Display Train who arrives at the route point in between 12AM to 12PM**

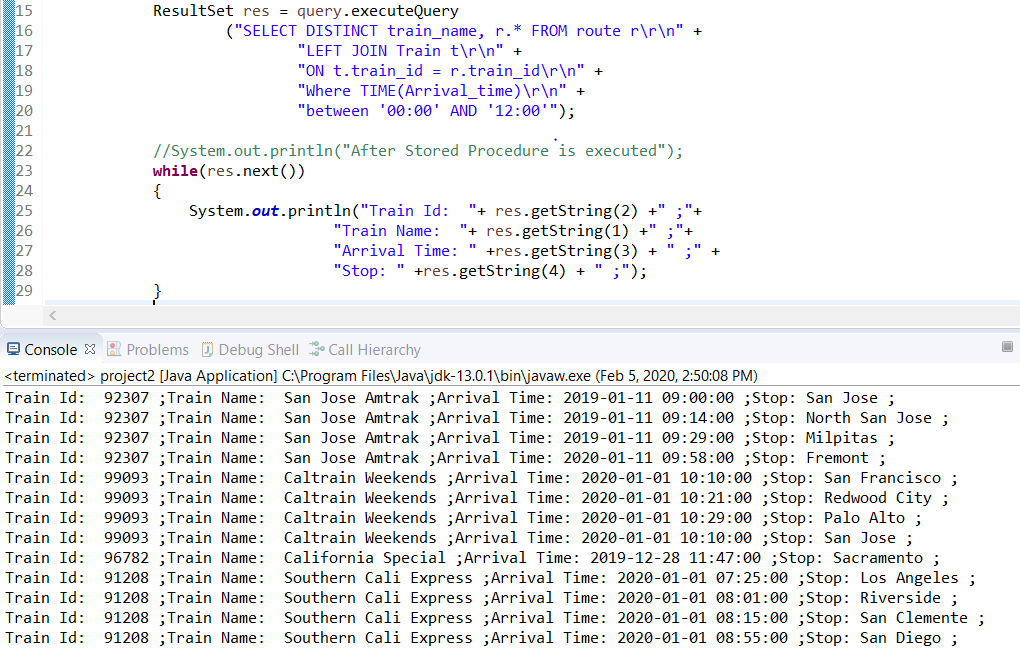
SELECT DISTINCT train\_name, r.\* FROM route r

LEFT JOIN Train t

ON t.train\_id = r.train\_id

Where TIME(Arrival\_time)

between '00:00' AND '12:00'



1. **Delete a Record From Route Table**





1. **Insertion of duplicate key (Constraint Violation)**

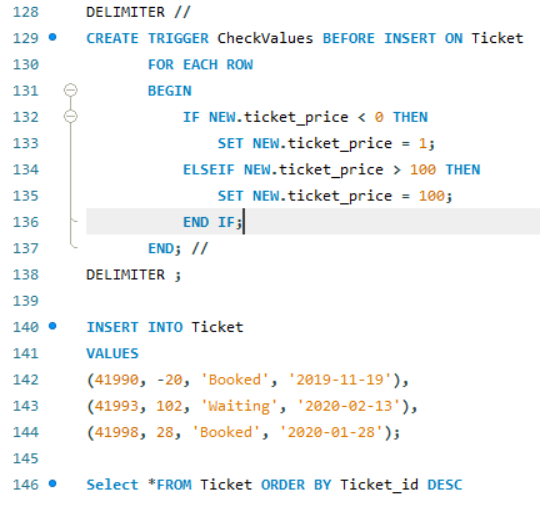


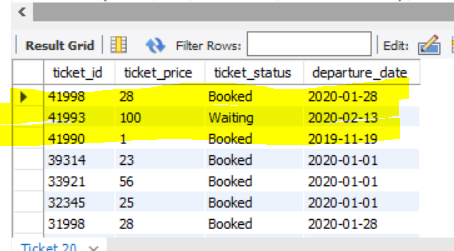


1. **Create Trigger**

Ticket Price cannot be less than 0

OR cannot be greater than 100

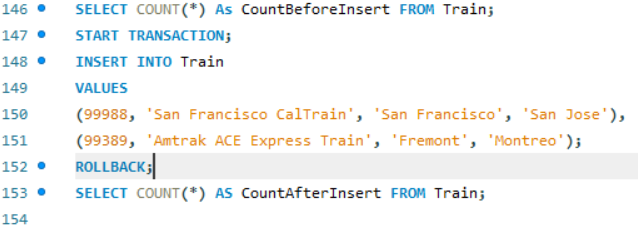


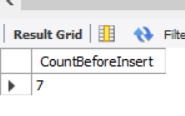


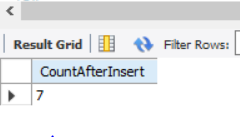
1. **Create Transaction**

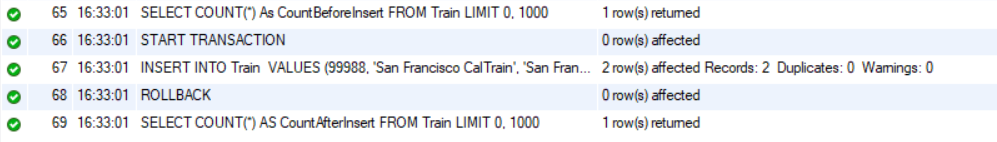
In this query we are creating a transaction, and then inserting data in the train table.

However, as we are immediately ROLLING BACK Transaction the values will not get stored in the train table.









**FILES FOR REFERENCES:**

****

****