PROJECT 2

ADVANCE DATABASE

GROUP 5

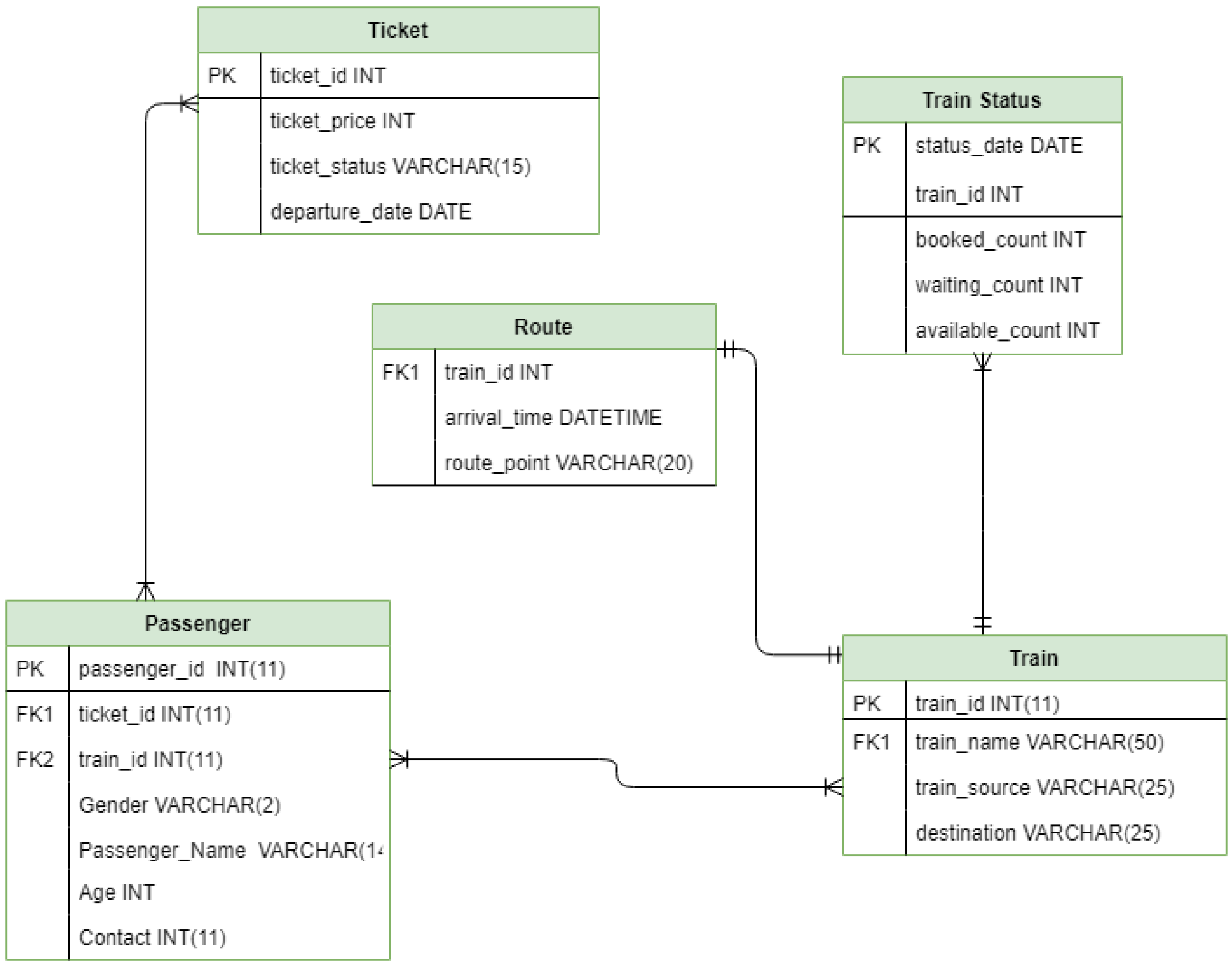
Vaishali Gupta	(00001588183)
Sarvesh Kulkarni	(00001588389)
Siddhant Kshatriya	(00001588464)
Thomas Francis	(00001561030)
Manas Sandhawani	(00001588394)

Abstract

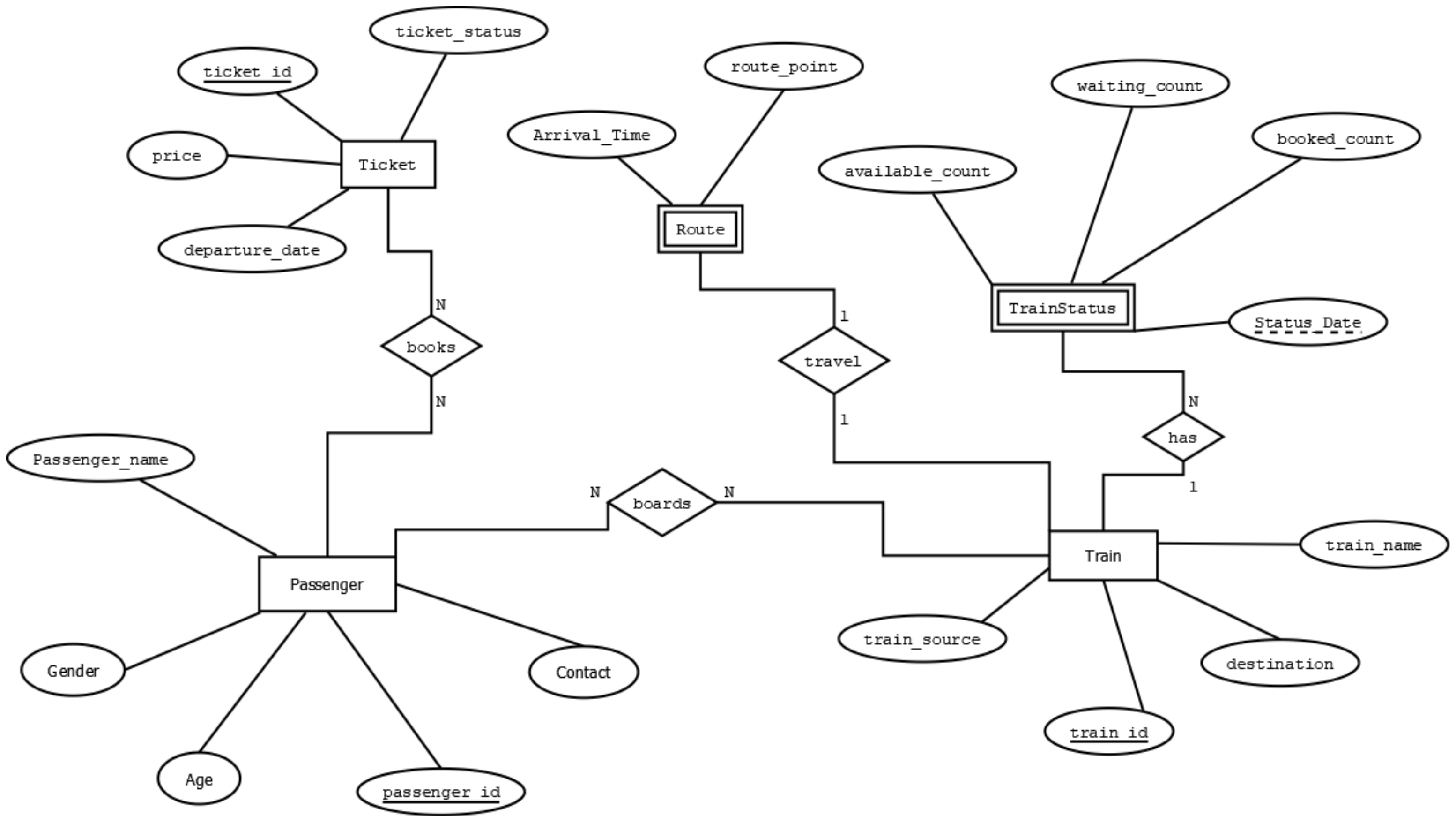
The project focuses on Train Management System which will allow the track of all train schedules, their route and passenger details. This will also allow the Train Administrator to manage, edit and add routes. Also, we can add new passenger and edit ticket details. This System simplifies the train management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


ticket_status
ticket_id
price
Ticket
departure_date
route_point
Arrival_Time
Route
waiting_count
booked_count
available_count
TrainStatus
Status_Date
N
books
N
1
travel
1
N
has
1
Passenger_name
N
boards
N
Passenger
Train
train_name
train_source
destination
train_id
Gender
Contact
Age
passenger_id

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

************************************************************************************************
```

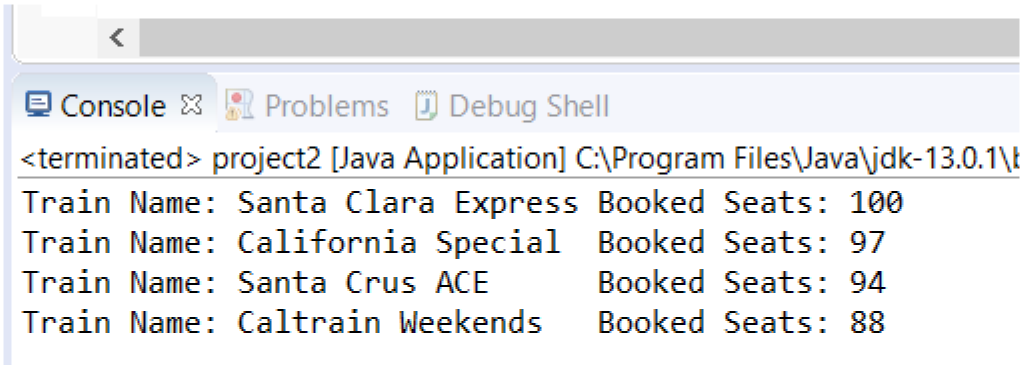
```
Select * from ticket
Ticket Id: 30010  ticket_price: 25 ticket status: Booked ticket status: 2020-01-01
Ticket Id: 30110  ticket_price: 35 ticket status: Waiting ticket status: 2020-01-01
Ticket Id: 30128  ticket_price: 5 ticket status: Booked ticket status: 2019-12-13
Ticket Id: 30157  ticket_price: 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**

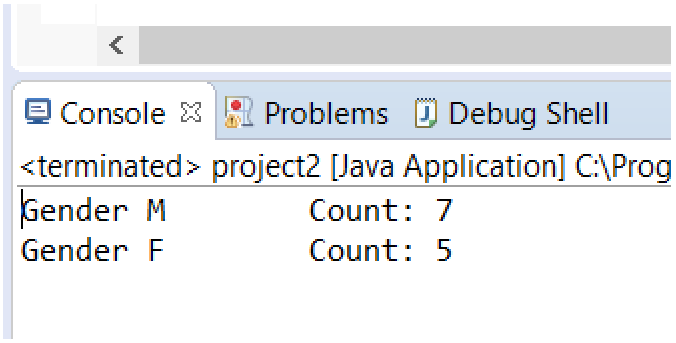
```
SELECT Train_name, booked_count FROM Train t
INNER JOIN TrainStatus ts ON
t.train_id = ts.train_id
ORDER BY Booked_count DESC limit 4;
```



```
<terminated> project2 [Java Application] C:\Program Files\Java\jdk-13.0.1\
Train Name: Santa Clara Express Booked Seats: 100
Train Name: California Special  Booked Seats: 97
Train Name: Santa Crus ACE      Booked Seats: 94
Train Name: Caltrain Weekends   Booked Seats: 88
```

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

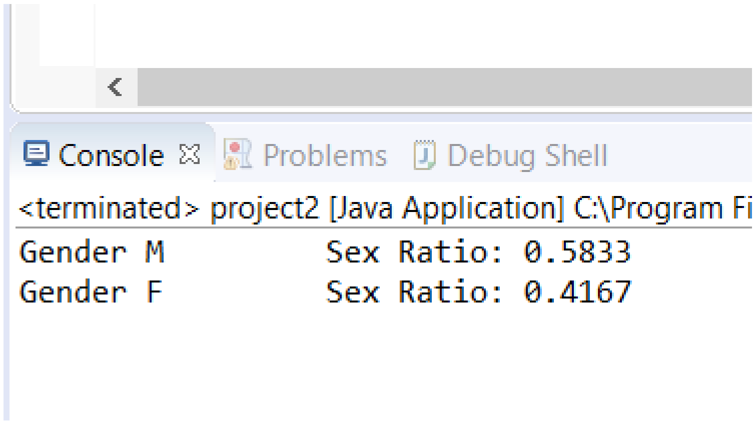
```
SELECT Gender, COUNT(passenger_ID) FROM passenger GROUP BY gender
```



```
<terminated> project2 [Java Application] C:\Prog
Gender M          Count: 7
Gender F          Count: 5
```

3. Ratio of Male to Female

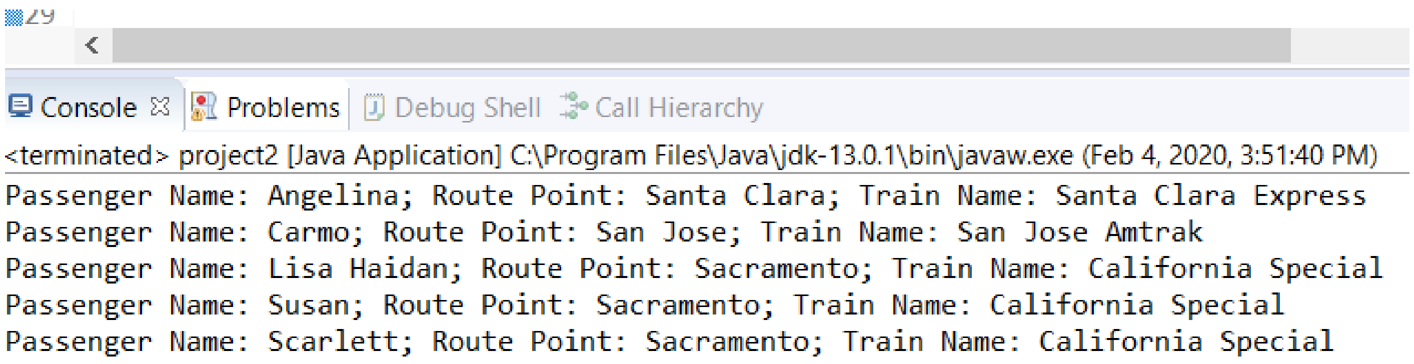
SELECT gender, Count(*) / (select count(*) FROM Passenger) AS Sex_Ratio FROM passenger
GROUP BY gender



```
<terminated> project2 [Java Application] C:\Program Fi
Gender M          Sex Ratio: 0.5833
Gender F          Sex Ratio: 0.4167
```

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

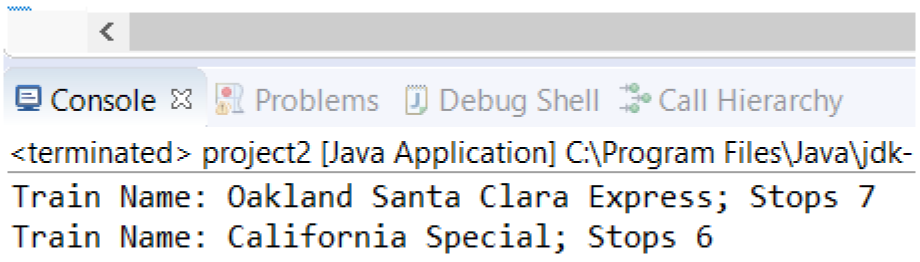
SELECT DISTINCT(passenger_name), route_point, train_name FROM passenger p
JOIN train t
ON t.train_id = p.train_id
JOIN route r
ON t.train_id = r.train_id
WHERE gender="F"
GROUP BY Passenger_NAme;



```
<terminated> project2 [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (Feb 4, 2020, 3:51:40 PM)
Passenger Name: Angelina; Route Point: Santa Clara; Train Name: Santa Clara Express
Passenger Name: Carmo; Route Point: San Jose; Train Name: San Jose Amtrak
Passenger Name: Lisa Haidan; Route Point: Sacramento; Train Name: California Special
Passenger Name: Susan; Route Point: Sacramento; Train Name: California Special
Passenger Name: Scarlett; Route Point: Sacramento; Train Name: California Special
```

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

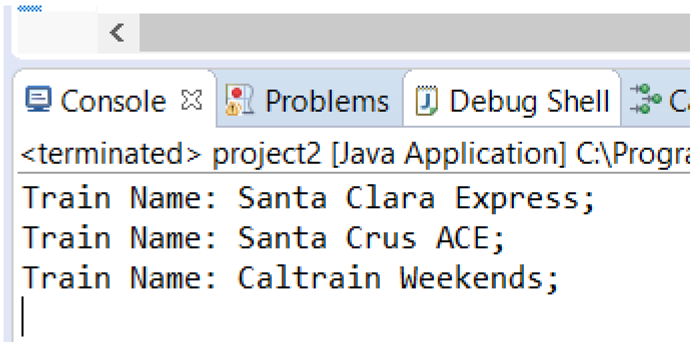
```
SELECT train_name, count(*)  Stops
FROM train T
JOIN
route r
ON r.train_id = t.train_id
GROUP BY r.train_id, Date(arrival_time)
HAVING Stops > 4
```



```
<terminated> project2 [Java Application] C:\Program Files\Java\jdk-
Train Name: Oakland Santa Clara Express; Stops 7
Train Name: California Special; Stops 6
```

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

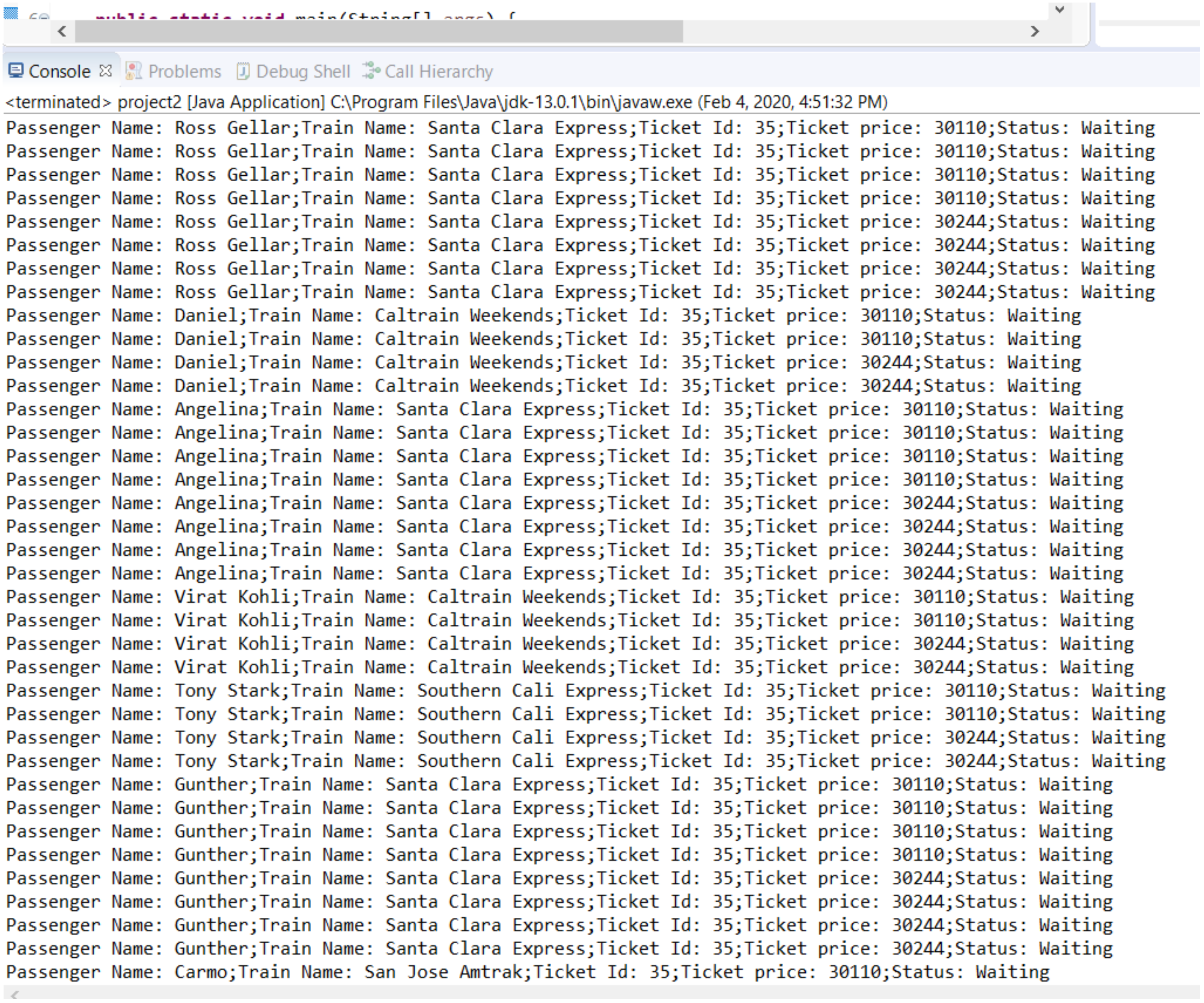
```
SELECT train_name FROM train
WHERE train_id IN
(SELECT route.train_id FROM route
WHERE route.route_point="San Francisco");
```



```
<terminated> project2 [Java Application] C:\Progra
Train Name: Santa Clara Express;
Train Name: Santa Crus ACE;
Train Name: Caltrain Weekends;
```

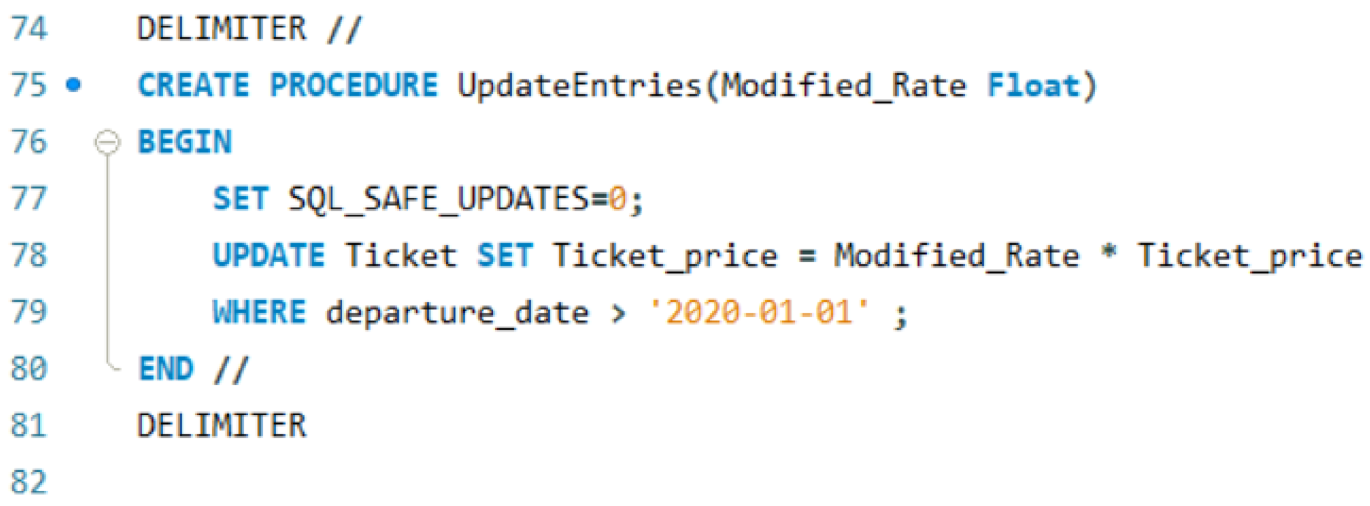
7. **Display Passenger and train details combined where ticket price is equal to 35 (4 Way Join)**

```
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
```



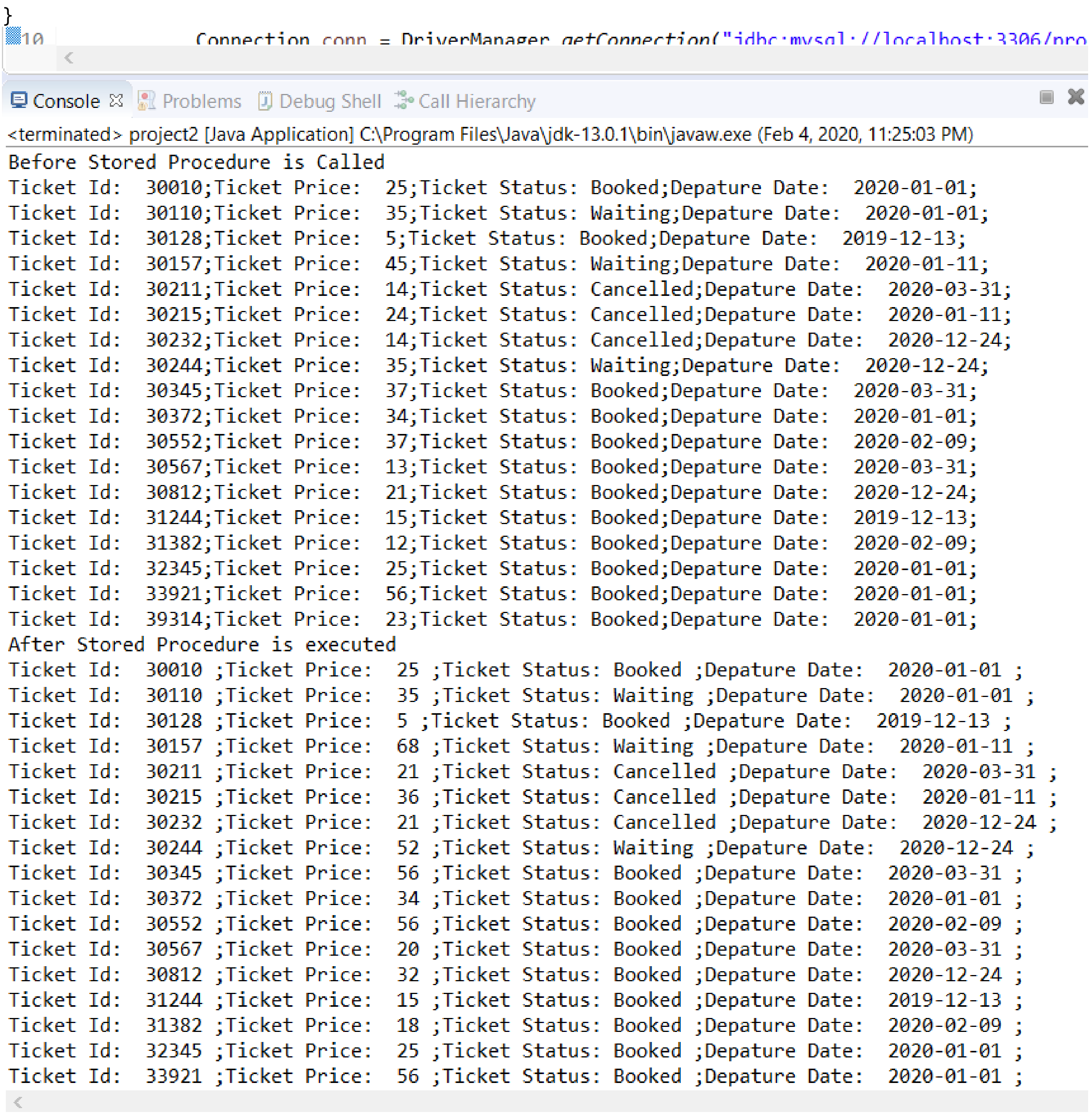
```
<terminated> project2 [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (Feb 4, 2020, 4:51:32 PM)
Passenger Name: Ross Gellar;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Ross Gellar;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Ross Gellar;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Ross Gellar;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Ross Gellar;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Ross Gellar;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Ross Gellar;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Ross Gellar;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Daniel;Train Name: Caltrain Weekends;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Daniel;Train Name: Caltrain Weekends;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Daniel;Train Name: Caltrain Weekends;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Daniel;Train Name: Caltrain Weekends;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Angelina;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Angelina;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Angelina;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Angelina;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Angelina;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Angelina;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Angelina;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Angelina;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Virat Kohli;Train Name: Caltrain Weekends;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Virat Kohli;Train Name: Caltrain Weekends;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Virat Kohli;Train Name: Caltrain Weekends;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Virat Kohli;Train Name: Caltrain Weekends;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Tony Stark;Train Name: Southern Cali Express;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Tony Stark;Train Name: Southern Cali Express;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Tony Stark;Train Name: Southern Cali Express;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Tony Stark;Train Name: Southern Cali Express;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Gunther;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Gunther;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Gunther;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Gunther;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30110;Status: Waiting
Passenger Name: Gunther;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Gunther;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Gunther;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Gunther;Train Name: Santa Clara Express;Ticket Id: 35;Ticket price: 30244;Status: Waiting
Passenger Name: Carmo;Train Name: San Jose Amtrak;Ticket Id: 35;Ticket price: 30110;Status: Waiting
```

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



```sql
DELIMITER //
CREATE PROCEDURE UpdateEntries(Modified_Rate Float)
BEGIN
    SET SQL_SAFE_UPDATES=0;
    UPDATE Ticket SET Ticket_price = Modified_Rate * Ticket_price
    WHERE departure_date > '2020-01-01' ;
END //
DELIMITER
```

```java
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();
		}
	}
```

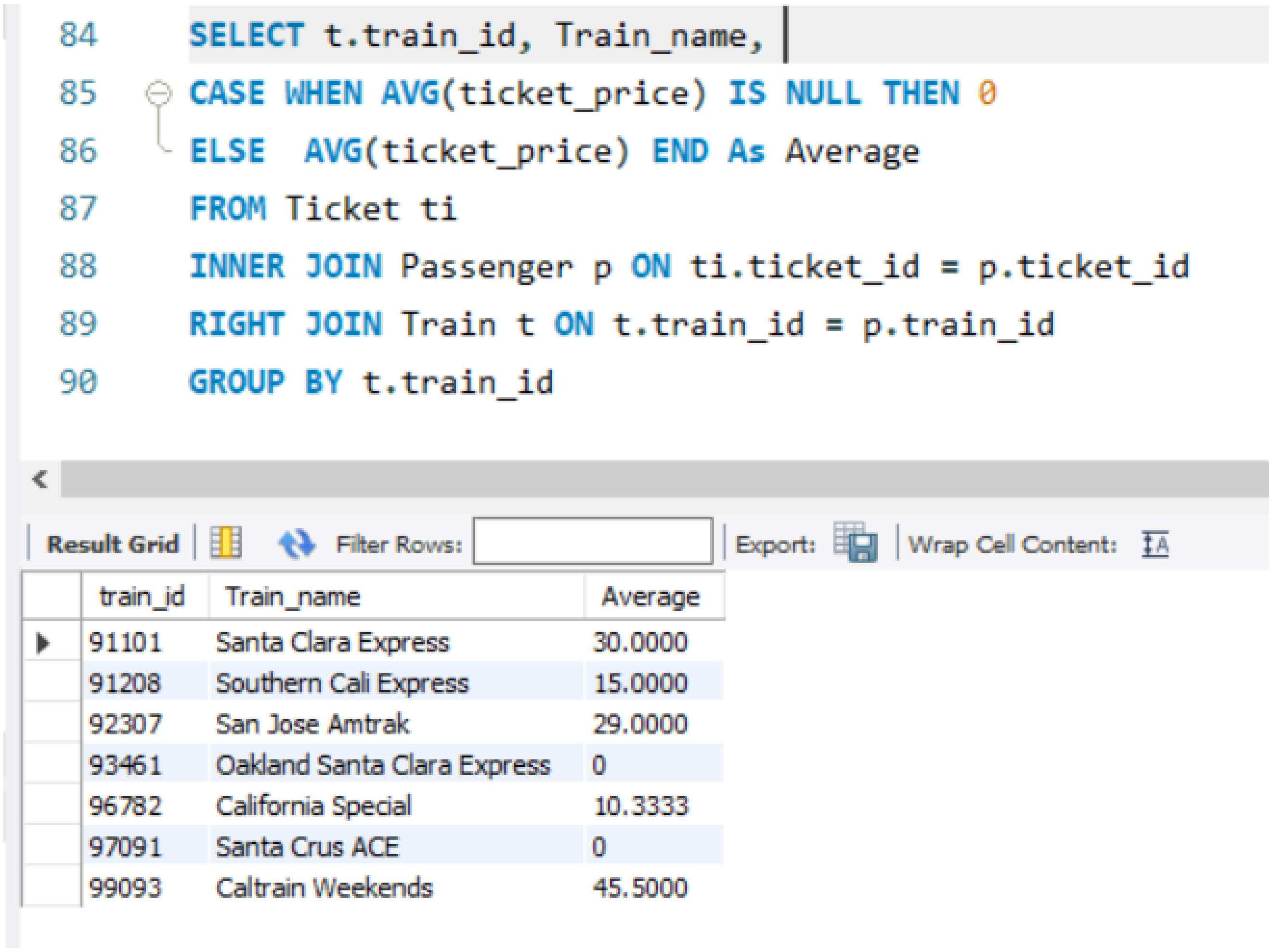


```
}
        Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/pro
```

Console ⊠ Problems Debug Shell Call Hierarchy

```
<terminated> project2 [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (Feb 4, 2020, 11:25:03 PM)
Before Stored Procedure is Called
Ticket Id:  30010;Ticket Price:  25;Ticket Status: Booked;Depature Date:  2020-01-01;
Ticket Id:  30110;Ticket Price:  35;Ticket Status: Waiting;Depature Date:  2020-01-01;
Ticket Id:  30128;Ticket Price:  5;Ticket Status: Booked;Depature Date:  2019-12-13;
Ticket Id:  30157;Ticket Price:  45;Ticket Status: Waiting;Depature Date:  2020-01-11;
Ticket Id:  30211;Ticket Price:  14;Ticket Status: Cancelled;Depature Date:  2020-03-31;
Ticket Id:  30215;Ticket Price:  24;Ticket Status: Cancelled;Depature Date:  2020-01-11;
Ticket Id:  30232;Ticket Price:  14;Ticket Status: Cancelled;Depature Date:  2020-12-24;
Ticket Id:  30244;Ticket Price:  35;Ticket Status: Waiting;Depature Date:  2020-12-24;
Ticket Id:  30345;Ticket Price:  37;Ticket Status: Booked;Depature Date:  2020-03-31;
Ticket Id:  30372;Ticket Price:  34;Ticket Status: Booked;Depature Date:  2020-01-01;
Ticket Id:  30552;Ticket Price:  37;Ticket Status: Booked;Depature Date:  2020-02-09;
Ticket Id:  30567;Ticket Price:  13;Ticket Status: Booked;Depature Date:  2020-03-31;
Ticket Id:  30812;Ticket Price:  21;Ticket Status: Booked;Depature Date:  2020-12-24;
Ticket Id:  31244;Ticket Price:  15;Ticket Status: Booked;Depature Date:  2019-12-13;
Ticket Id:  31382;Ticket Price:  12;Ticket Status: Booked;Depature Date:  2020-02-09;
Ticket Id:  32345;Ticket Price:  25;Ticket Status: Booked;Depature Date:  2020-01-01;
Ticket Id:  33921;Ticket Price:  56;Ticket Status: Booked;Depature Date:  2020-01-01;
Ticket Id:  39314;Ticket Price:  23;Ticket Status: Booked;Depature Date:  2020-01-01;
After Stored Procedure is executed
Ticket Id:  30010 ;Ticket Price:  25 ;Ticket Status: Booked ;Depature Date:  2020-01-01 ;
Ticket Id:  30110 ;Ticket Price:  35 ;Ticket Status: Waiting ;Depature Date:  2020-01-01 ;
Ticket Id:  30128 ;Ticket Price:  5 ;Ticket Status: Booked ;Depature Date:  2019-12-13 ;
Ticket Id:  30157 ;Ticket Price:  68 ;Ticket Status: Waiting ;Depature Date:  2020-01-11 ;
Ticket Id:  30211 ;Ticket Price:  21 ;Ticket Status: Cancelled ;Depature Date:  2020-03-31 ;
Ticket Id:  30215 ;Ticket Price:  36 ;Ticket Status: Cancelled ;Depature Date:  2020-01-11 ;
Ticket Id:  30232 ;Ticket Price:  21 ;Ticket Status: Cancelled ;Depature Date:  2020-12-24 ;
Ticket Id:  30244 ;Ticket Price:  52 ;Ticket Status: Waiting ;Depature Date:  2020-12-24 ;
Ticket Id:  30345 ;Ticket Price:  56 ;Ticket Status: Booked ;Depature Date:  2020-03-31 ;
Ticket Id:  30372 ;Ticket Price:  34 ;Ticket Status: Booked ;Depature Date:  2020-01-01 ;
Ticket Id:  30552 ;Ticket Price:  56 ;Ticket Status: Booked ;Depature Date:  2020-02-09 ;
Ticket Id:  30567 ;Ticket Price:  20 ;Ticket Status: Booked ;Depature Date:  2020-03-31 ;
Ticket Id:  30812 ;Ticket Price:  32 ;Ticket Status: Booked ;Depature Date:  2020-12-24 ;
Ticket Id:  31244 ;Ticket Price:  15 ;Ticket Status: Booked ;Depature Date:  2019-12-13 ;
Ticket Id:  31382 ;Ticket Price:  18 ;Ticket Status: Booked ;Depature Date:  2020-02-09 ;
Ticket Id:  32345 ;Ticket Price:  25 ;Ticket Status: Booked ;Depature Date:  2020-01-01 ;
Ticket Id:  33921 ;Ticket Price:  56 ;Ticket Status: Booked ;Depature Date:  2020-01-01 ;
```

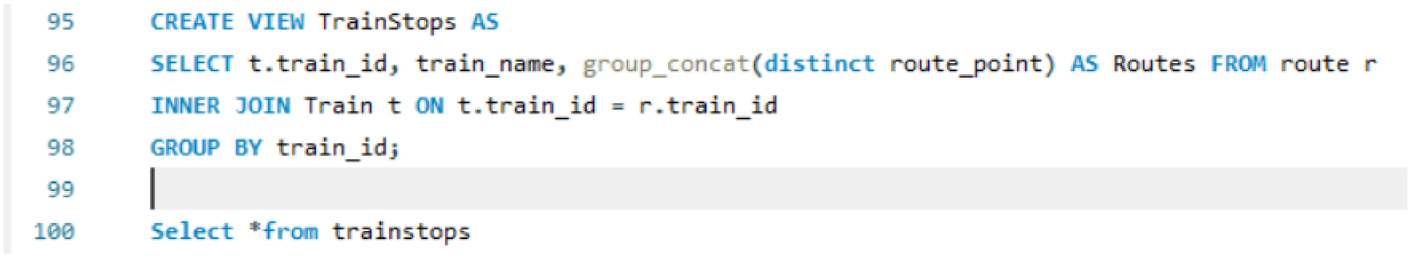
9. Display Average Ticket Price for all the trains, if no tickets are present for train show as 0



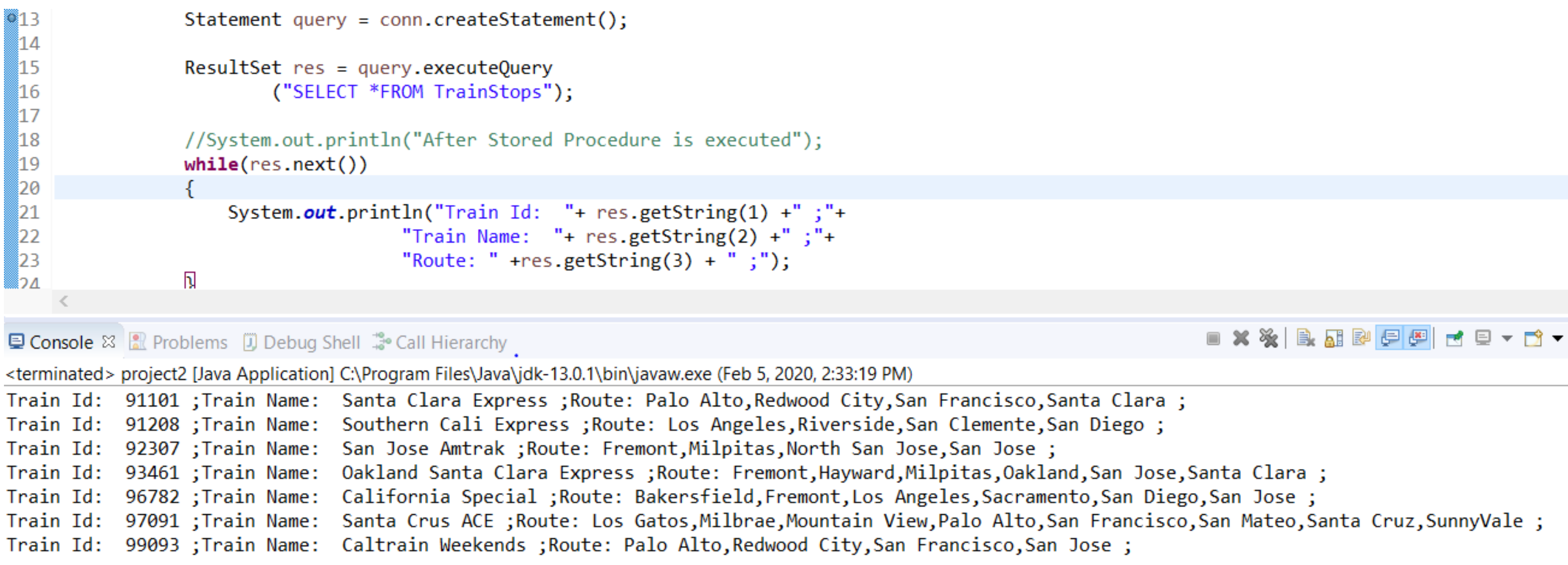
```sql
SELECT t.train_id, Train_name,
CASE WHEN AVG(ticket_price) IS NULL THEN 0
ELSE  AVG(ticket_price) END As Average
FROM Ticket ti
INNER JOIN Passenger p ON ti.ticket_id = p.ticket_id
RIGHT JOIN Train t ON t.train_id = p.train_id
GROUP BY t.train_id
```

train_id	Train_name	Average
91101	Santa Clara Express	30.0000
91208	Southern Cali Express	15.0000
92307	San Jose Amtrak	29.0000
93461	Oakland Santa Clara Express	0
96782	California Special	10.3333
97091	Santa Crus ACE	0
99093	Caltrain Weekends	45.5000

10. Create View of Train Name and their Routes.



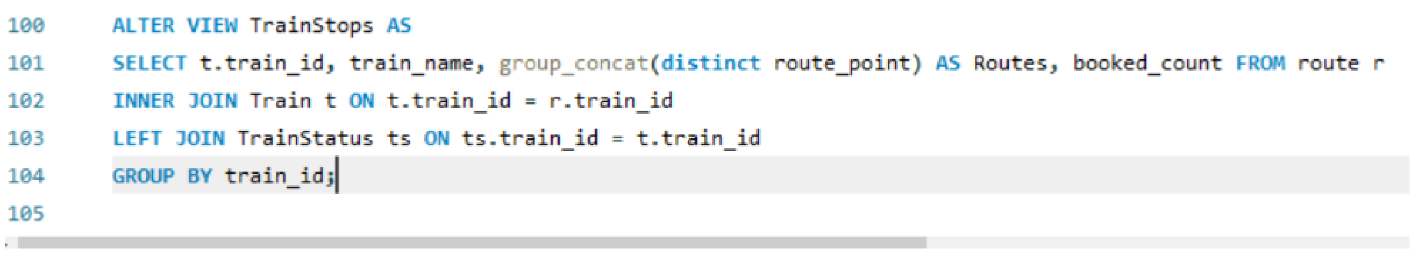
```sql
CREATE VIEW TrainStops AS
SELECT t.train_id, train_name, group_concat(distinct route_point) AS Routes FROM route r
INNER JOIN Train t ON t.train_id = r.train_id
GROUP BY train_id;

Select *from trainstops
```



```
        Statement query = conn.createStatement();

        ResultSet res = query.executeQuery
                ("SELECT *FROM TrainStops");

        //System.out.println("After Stored Procedure is executed");
        while(res.next())
        {
            System.out.println("Train Id:  "+ res.getString(1) +" ;"+
                            "Train Name:  "+ res.getString(2) +" ;"+
                            "Route: " +res.getString(3) + " ;");
        }
```

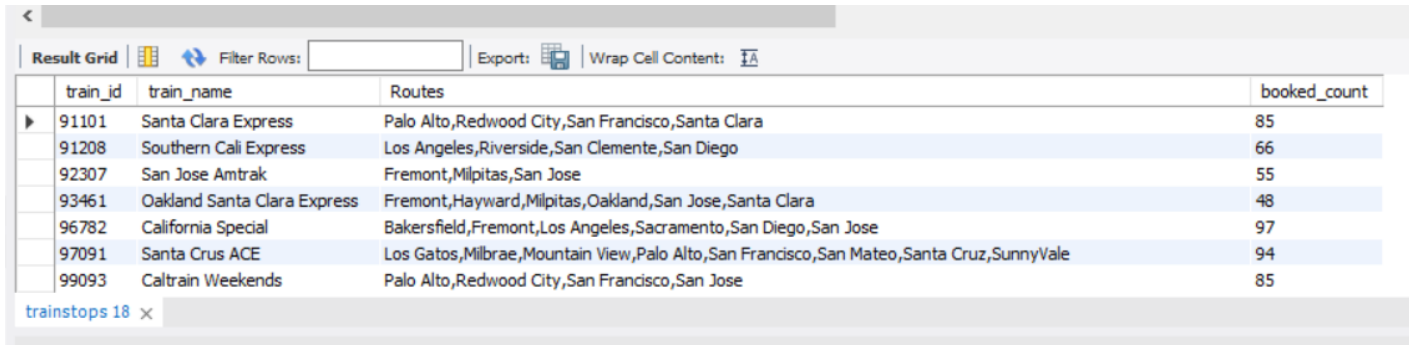
```
<terminated> project2 [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (Feb 5, 2020, 2:33:19 PM)
Train Id:  91101 ;Train Name:  Santa Clara Express ;Route: Palo Alto,Redwood City,San Francisco,Santa Clara ;
Train Id:  91208 ;Train Name:  Southern Cali Express ;Route: Los Angeles,Riverside,San Clemente,San Diego ;
Train Id:  92307 ;Train Name:  San Jose Amtrak ;Route: Fremont,Milpitas,North San Jose,San Jose ;
Train Id:  93461 ;Train Name:  Oakland Santa Clara Express ;Route: Fremont,Hayward,Milpitas,Oakland,San Jose,Santa Clara ;
Train Id:  96782 ;Train Name:  California Special ;Route: Bakersfield,Fremont,Los Angeles,Sacramento,San Diego,San Jose ;
Train Id:  97091 ;Train Name:  Santa Crus ACE ;Route: Los Gatos,Milbrae,Mountain View,Palo Alto,San Francisco,San Mateo,Santa Cruz,SunnyVale ;
Train Id:  99093 ;Train Name:  Caltrain Weekends ;Route: Palo Alto,Redwood City,San Francisco,San Jose ;
```

11. Update View



```
ALTER VIEW TrainStops AS
SELECT t.train_id, train_name, group_concat(distinct route_point) AS Routes, booked_count FROM route r
INNER JOIN Train t ON t.train_id = r.train_id
LEFT JOIN TrainStatus ts ON ts.train_id = t.train_id
GROUP BY train_id;
```

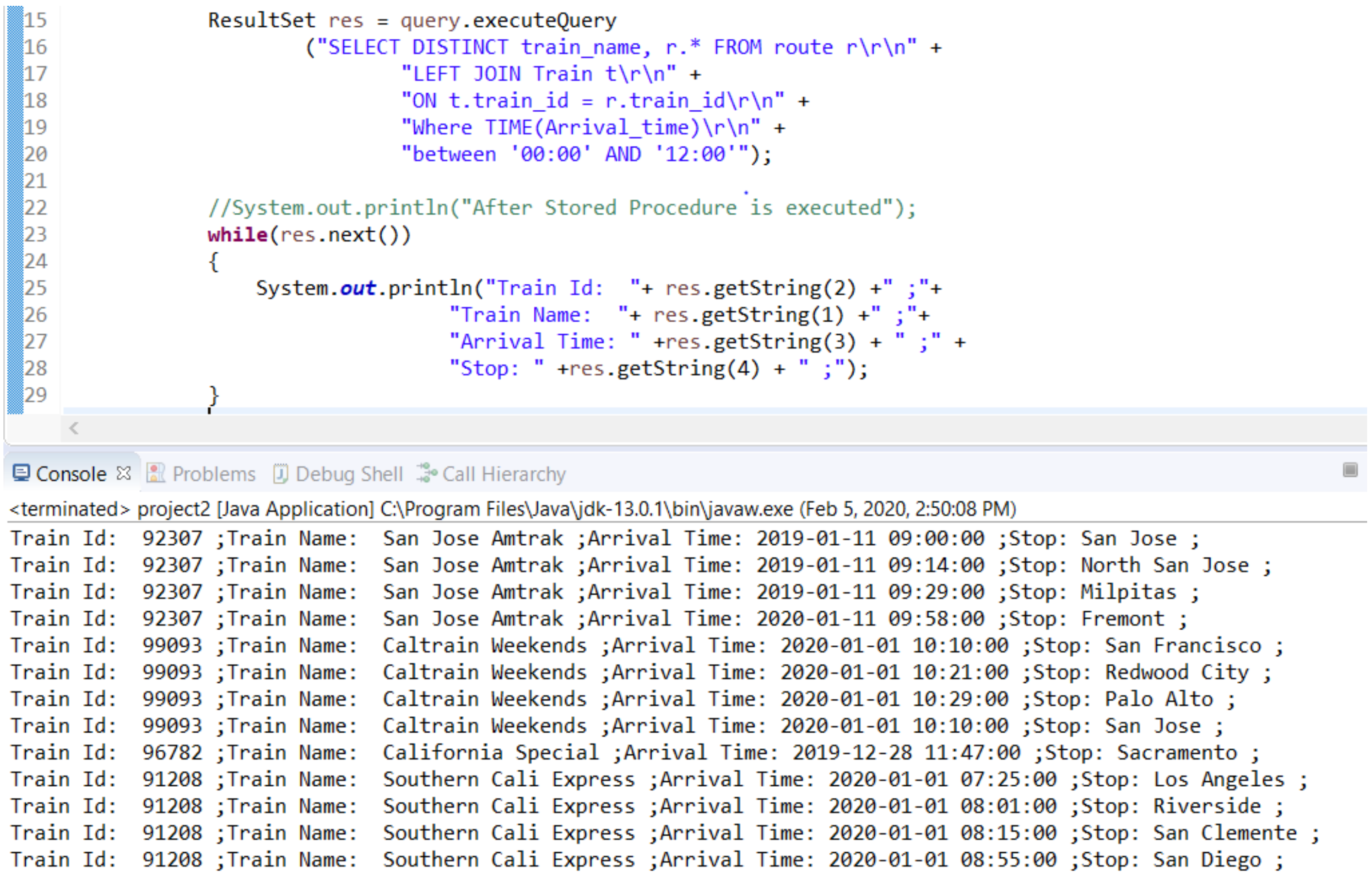
SELECT *FROM TrainStops



train_id	train_name	Routes	booked_count
91101	Santa Clara Express	Palo Alto,Redwood City,San Francisco,Santa Clara	85
91208	Southern Cali Express	Los Angeles,Riverside,San Clemente,San Diego	66
92307	San Jose Amtrak	Fremont,Milpitas,San Jose	55
93461	Oakland Santa Clara Express	Fremont,Hayward,Milpitas,Oakland,San Jose,Santa Clara	48
96782	California Special	Bakersfield,Fremont,Los Angeles,Sacramento,San Diego,San Jose	97
97091	Santa Crus ACE	Los Gatos,Milbrae,Mountain View,Palo Alto,San Francisco,San Mateo,Santa Cruz,SunnyVale	94
99093	Caltrain Weekends	Palo Alto,Redwood City,San Francisco,San Jose	85

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

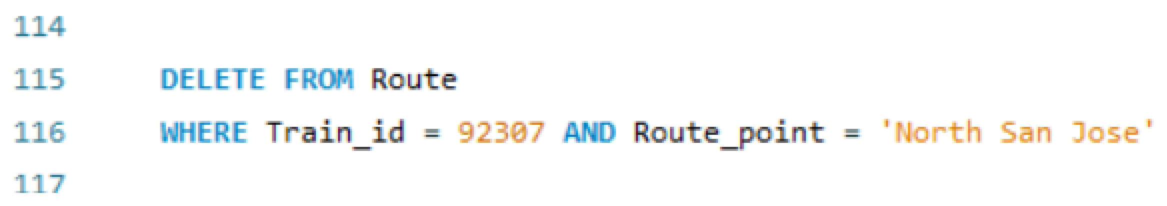
```
SELECT DISTINCT train_name, r.* FROM route r
LEFT JOIN Train t
ON t.train_id = r.train_id
Where TIME(Arrival_time)
between '00:00' AND '12:00'
```



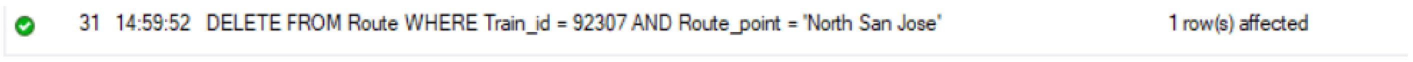
```
        ResultSet res = query.executeQuery
                ("SELECT DISTINCT train_name, r.* FROM route r\r\n" +
                        "LEFT JOIN Train t\r\n" +
                        "ON t.train_id = r.train_id\r\n" +
                        "Where TIME(Arrival_time)\r\n" +
                        "between '00:00' AND '12:00'");

        //System.out.println("After Stored Procedure is executed");
        while(res.next())
        {
            System.out.println("Train Id:  "+ res.getString(2) +" ;"+
                            "Train Name:  "+ res.getString(1) +" ;"+
                            "Arrival Time: " +res.getString(3) + " ;" +
                            "Stop: " +res.getString(4) + " ;");
        }
```

```
<terminated> project2 [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (Feb 5, 2020, 2:50:08 PM)
Train Id:  92307 ;Train Name:  San Jose Amtrak ;Arrival Time: 2019-01-11 09:00:00 ;Stop: San Jose ;
Train Id:  92307 ;Train Name:  San Jose Amtrak ;Arrival Time: 2019-01-11 09:14:00 ;Stop: North San Jose ;
Train Id:  92307 ;Train Name:  San Jose Amtrak ;Arrival Time: 2019-01-11 09:29:00 ;Stop: Milpitas ;
Train Id:  92307 ;Train Name:  San Jose Amtrak ;Arrival Time: 2020-01-11 09:58:00 ;Stop: Fremont ;
Train Id:  99093 ;Train Name:  Caltrain Weekends ;Arrival Time: 2020-01-01 10:10:00 ;Stop: San Francisco ;
Train Id:  99093 ;Train Name:  Caltrain Weekends ;Arrival Time: 2020-01-01 10:21:00 ;Stop: Redwood City ;
Train Id:  99093 ;Train Name:  Caltrain Weekends ;Arrival Time: 2020-01-01 10:29:00 ;Stop: Palo Alto ;
Train Id:  99093 ;Train Name:  Caltrain Weekends ;Arrival Time: 2020-01-01 10:10:00 ;Stop: San Jose ;
Train Id:  96782 ;Train Name:  California Special ;Arrival Time: 2019-12-28 11:47:00 ;Stop: Sacramento ;
Train Id:  91208 ;Train Name:  Southern Cali Express ;Arrival Time: 2020-01-01 07:25:00 ;Stop: Los Angeles ;
Train Id:  91208 ;Train Name:  Southern Cali Express ;Arrival Time: 2020-01-01 08:01:00 ;Stop: Riverside ;
Train Id:  91208 ;Train Name:  Southern Cali Express ;Arrival Time: 2020-01-01 08:15:00 ;Stop: San Clemente ;
Train Id:  91208 ;Train Name:  Southern Cali Express ;Arrival Time: 2020-01-01 08:55:00 ;Stop: San Diego ;
```

13. Delete a Record From Route Table

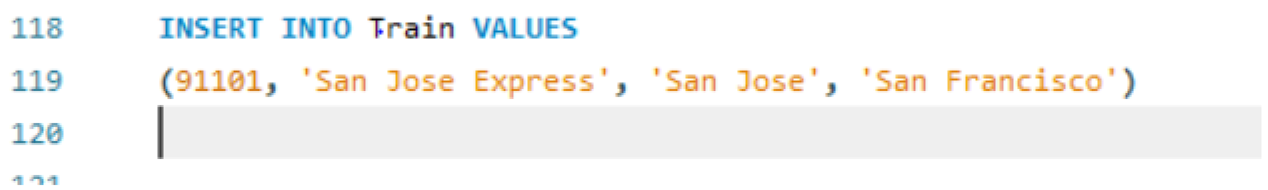


```
DELETE FROM Route
WHERE Train_id = 92307 AND Route_point = 'North San Jose'
```

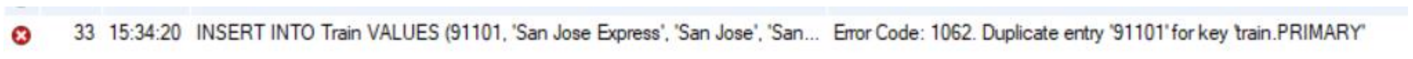


```
31  14:59:52  DELETE FROM Route WHERE Train_id = 92307 AND Route_point = 'North San Jose'        1 row(s) affected
```

14. Insertion of duplicate key (Constraint Violation)



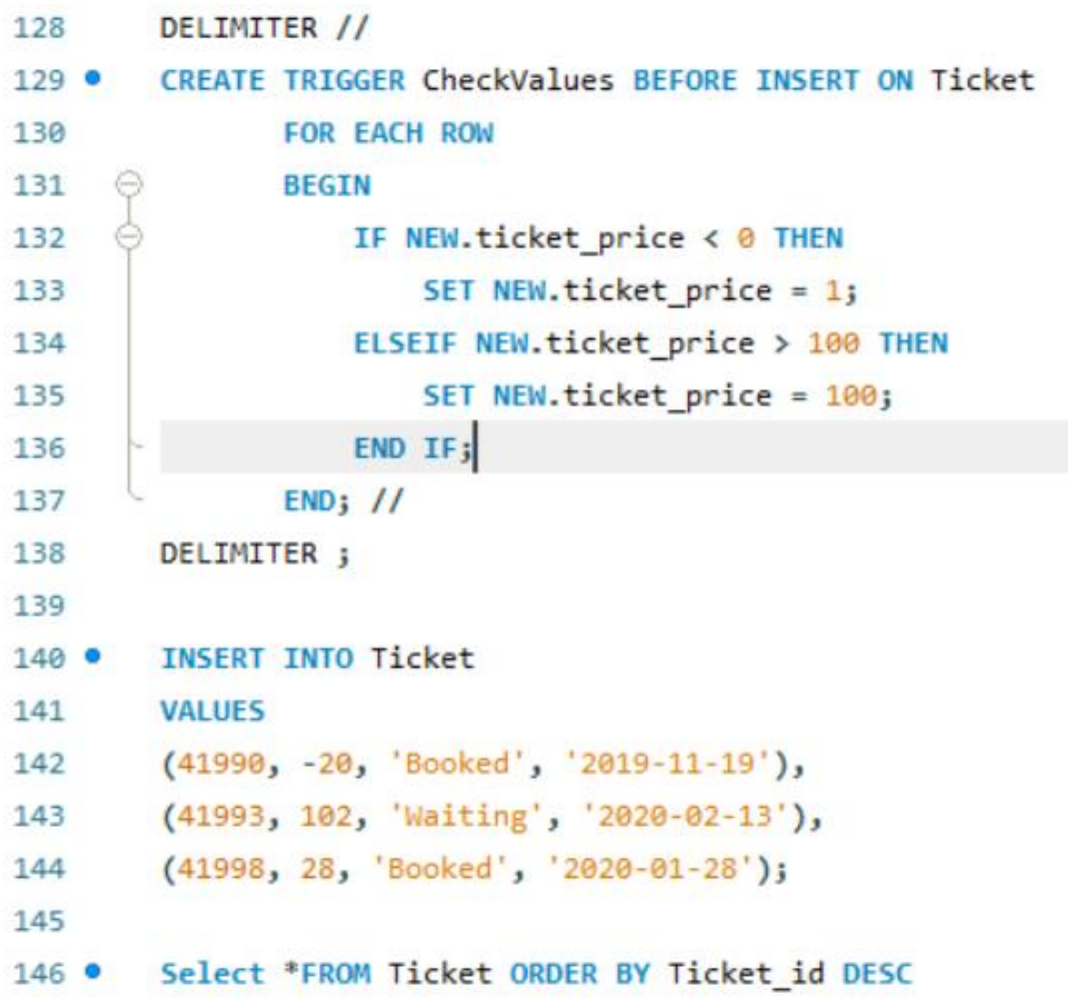
```
INSERT INTO Train VALUES
(91101, 'San Jose Express', 'San Jose', 'San Francisco')
```



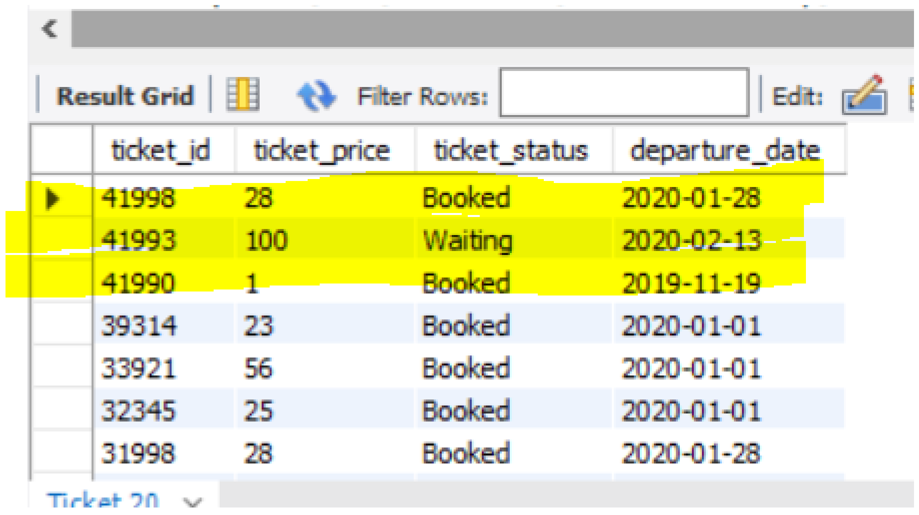
```
33  15:34:20  INSERT INTO Train VALUES (91101, 'San Jose Express', 'San Jose', 'San...   Error Code: 1062. Duplicate entry '91101' for key 'train.PRIMARY'
```

15. Create Trigger

Ticket Price cannot be less than 0
OR cannot be greater than 100



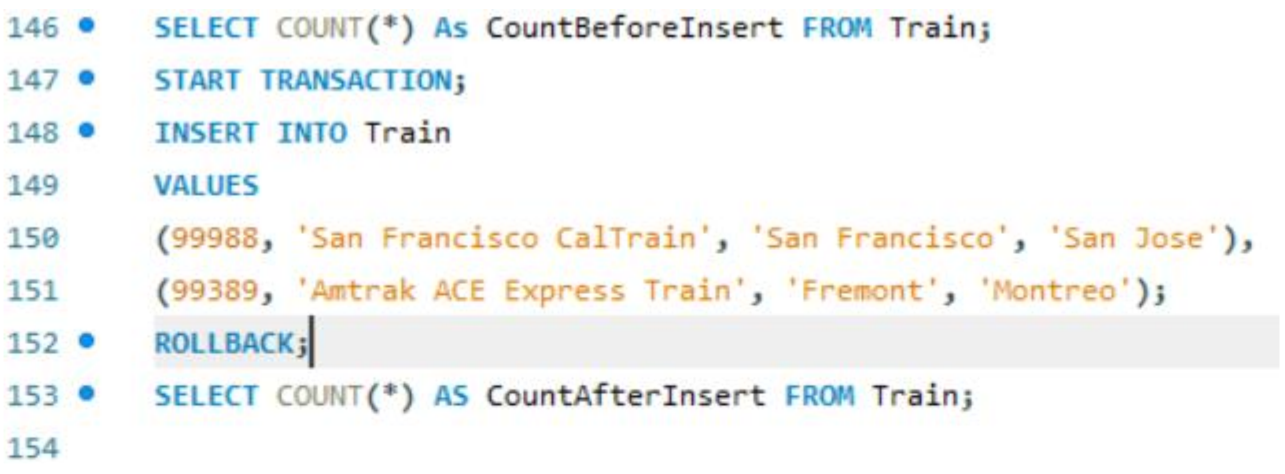
```
DELIMITER //
CREATE TRIGGER CheckValues BEFORE INSERT ON Ticket
        FOR EACH ROW
        BEGIN
            IF NEW.ticket_price < 0 THEN
                SET NEW.ticket_price = 1;
            ELSEIF NEW.ticket_price > 100 THEN
                SET NEW.ticket_price = 100;
            END IF;
        END; //
DELIMITER ;

INSERT INTO Ticket
VALUES
(41990, -20, 'Booked', '2019-11-19'),
(41993, 102, 'Waiting', '2020-02-13'),
(41998, 28, 'Booked', '2020-01-28');

Select *FROM Ticket ORDER BY Ticket_id DESC
```



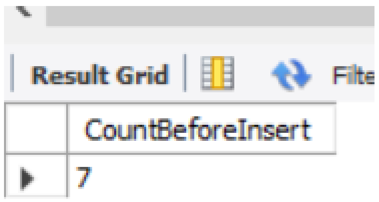
ticket_id	ticket_price	ticket_status	departure_date
41998	28	Booked	2020-01-28
41993	100	Waiting	2020-02-13
41990	1	Booked	2019-11-19
39314	23	Booked	2020-01-01
33921	56	Booked	2020-01-01
32345	25	Booked	2020-01-01
31998	28	Booked	2020-01-28

16. Create Transaction

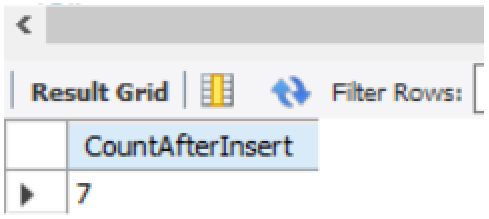
In this query we are creating a transaction, and then inserting data in the train table.
However, as we are immediately ROLLING BACK Transaction the values will not get stored in the train table.



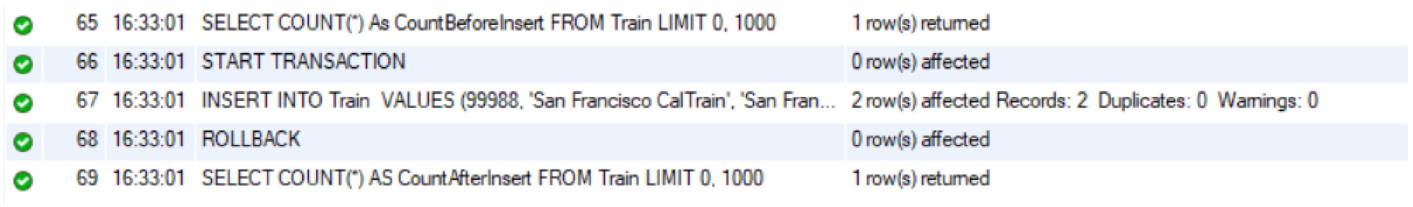
```sql
SELECT COUNT(*) As CountBeforeInsert FROM Train;
START TRANSACTION;
INSERT INTO Train
VALUES
(99988, 'San Francisco CalTrain', 'San Francisco', 'San Jose'),
(99389, 'Amtrak ACE Express Train', 'Fremont', 'Montreo');
ROLLBACK;
SELECT COUNT(*) AS CountAfterInsert FROM Train;
```



	CountBeforeInsert
▶	7



	CountAfterInsert
▶	7



```
65  16:33:01  SELECT COUNT(*) As CountBeforeInsert FROM Train LIMIT 0, 1000               1 row(s) returned
66  16:33:01  START TRANSACTION                                                           0 row(s) affected
67  16:33:01  INSERT INTO Train  VALUES (99988, 'San Francisco CalTrain', 'San Fran...     2 row(s) affected Records: 2  Duplicates: 0  Warnings: 0
68  16:33:01  ROLLBACK                                                                    0 row(s) affected
69  16:33:01  SELECT COUNT(*) AS CountAfterInsert FROM Train LIMIT 0, 1000                1 row(s) returned
```

FILES FOR REFERENCES:

Code-Project2.zip   projectJDBC.zip

P2InsertDataQueries.sql   P2CustomeQueries.sql   P2CreateTableQueries.sql