PES UNIVERSITY
Department of Computer Science & Engineering

DBMS - UE20CS301

Mini Project

FERRY MANAGEMENT SYSTEM

Submitted to:

Dr. Geetha D
Associate Professor

Submitted By:

Name: Shruthika Anand
SRN: PES2UG20CS334
V Semester
Section _F

Table of Contents

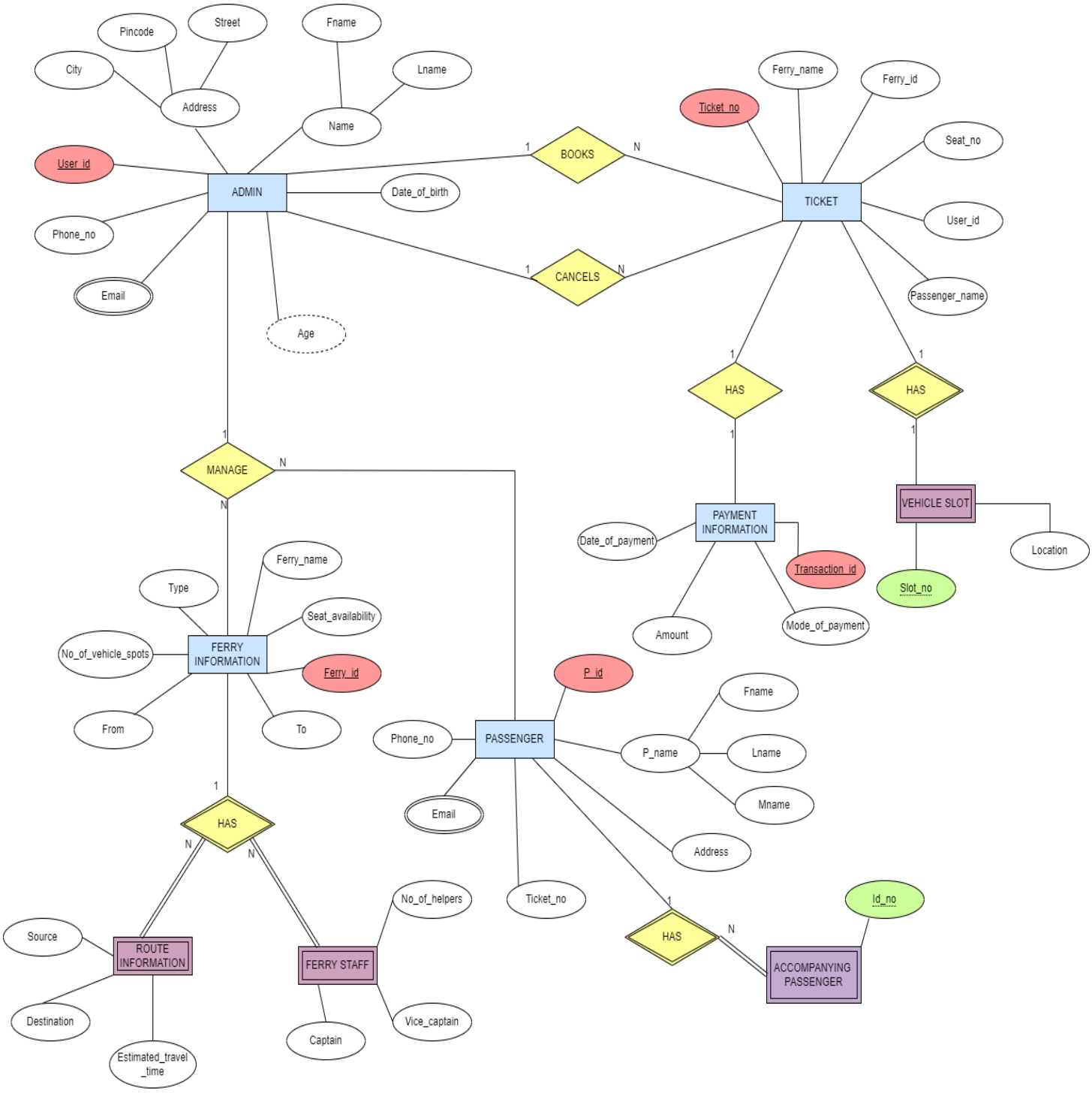
1. Short Description and Scope of the Project

Ferries provide vital connections which enable all types of journeys, whether that be local trips to work, study, visit others, or access amenities, or longer trips for business or tourism. Ferries also facilitate trade within and across national borders. The global ferry industry has a major impact on the economy and employment.

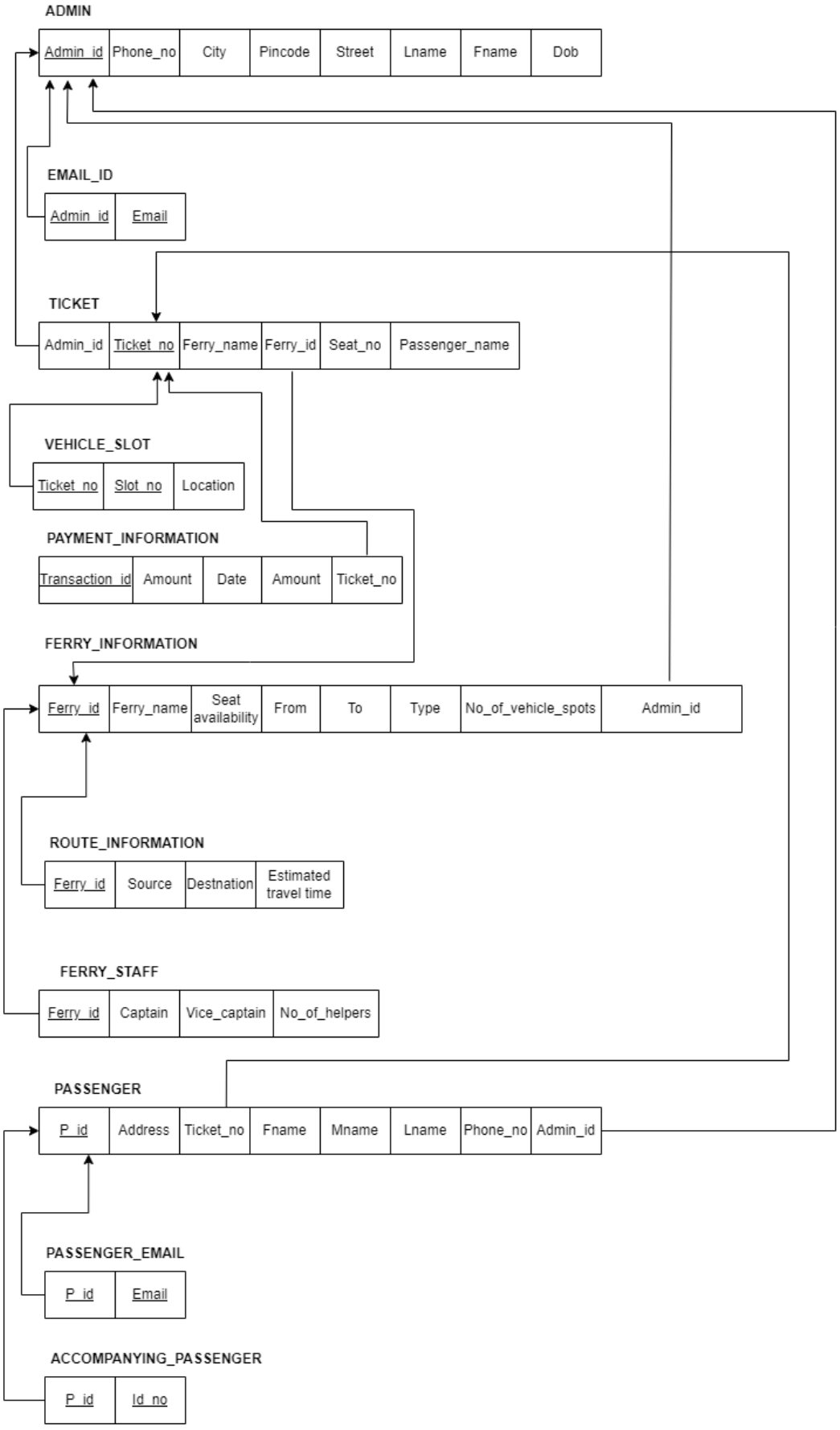
This project aims at building a simple ferry management system. The database will include information about ferry ride details, ticket details, passenger details, vehicle parking slots available and employee details and their role in the system.

The scope of this project will be a simplified ferry management system which will be easy to use by both employees and customers. Employees will be able to keep track of all the ferries and staff working on them. Customers will be able to view details of ferry rides, available tickets and reserve tickets.

2. ER Diagram


Pincode
Street
Fname
City
Lname
Address
Name
User_id
ADMIN
Date_of_birth
Phone_no
Email
Age
BOOKS
CANCELS
Ferry_name
Ferry_id
Ticket_no
Seat_no
TICKET
User_id
Passenger_name
HAS
HAS
MANAGE
PAYMENT INFORMATION
Date_of_payment
Transaction_id
Amount
Mode_of_payment
VEHICLE SLOT
Location
Slot_no
Type
Ferry_name
Seat_availability
No_of_vehicle_spots
FERRY INFORMATION
Ferry_id
From
To
P_id
Fname
Phone_no
PASSENGER
P_name
Lname
Mname
Email
Address
HAS
Ticket_no
Id_no
HAS
ACCOMPANYING PASSENGER
Source
ROUTE INFORMATION
Destination
Estimated_travel_time
FERRY STAFF
No_of_helpers
Captain
Vice_captain

3. Relational Schema



4. DDL statements - Building the database

CREATE TABLE ADMIN(Admin_id int, Fname varchar(30), Lname varchar(30), DOB date, Phone_no int, Email varchar(40), City varchar(30), Pincode int, Street varchar(40), PRIMARY KEY(Admin_id));

CREATE TABLE FERRY_INFO(Ferry_id varchar(20), Ferry_name varchar(35), Seat_availability int, source varchar(40), dest varchar(40), Vehicle_spots int, Type_of_ferry varchar(20), PRIMARY KEY(Ferry_id));

CREATE TABLE TICKET(Ticket_no int, Ferry_name varchar(35) UNIQUE, Ferry_id varchar(20) UNIQUE , Seat_no varchar(10), Admin_id int, Passenger_name varchar(30), PRIMARY KEY(Ticket_no), FOREIGN KEY(Admin_id) REFERENCES ADMIN(Admin_id));

CREATE TABLE PAYMENT(Trans_id int, Amount int, Date_trans date, Mode_of_payment varchar(20), Ticket_no int, PRIMARY KEY(Trans_id), FOREIGN KEY(Ticket_no) REFERENCES Ticket(Ticket_no));

CREATE TABLE PASSENGER(P_id int, Fname varchar(30), Mname varchar(30), Lnamme varchar(30), Phone_no int, Address varchar(50), Email varchar(40), Admin_id int, PRIMARY KEY(P_id), FOREIGN KEY(Admin_id) REFERENCES ADMIN(Admin_id));

CREATE TABLE ROUTE_INFO(Source varchar(20), Dest varchar(20), Travel_time int, Ferry_id varchar(20) UNIQUE);

CREATE TABLE FERRY_STAFF(Captain varchar(40), Vice_captain varchar(40), Helpers int, Ferry_id varchar(20) UNIQUE);

CREATE TABLE VEHICLE_SLOT(Slot_no int UNIQUE, Location varchar(25), Ticket_no int, FOREIGN KEY(Ticket_no) REFERENCES TICKET(Ticket_no));

CREATE TABLE ACCOMPANY(Id int, P_id int, PRIMARY KEY(Id), FOREIGN KEY(P_id) REFERENCES PASSENGER(P_id));

5. Populating the Database

```
INSERT INTO
ADMIN(Admin_id,Fname,Lname,DOB,Phone_no,Email,City,Pincode,Street)
VALUES
(2,'Riya','Jha','2002-12-1',12234567,'riya@gmail.com','Chennai',560078,'KK nagar'),
(3,'Arun','Vishwakumar','2000-2-
5',82635237,'arun@gmail.com','Bangalore',560078,'Whitefield'),
(4,'Harry','Styles','1998-12-
21',91726536,'style101@gmail.com','Chennai',901038,'Marina'),
(5,'Zayn','Malik','1980-12-
1',12211567,'zma@gmail.com','Hyderabad',510098,'Whitefield'),
(6,'Louis','Phillip','2002-11-6',126876567,'lp89@gmail.com','Chennai',590078,'Kasi
street'),
(7,'Liam','Payne','1999-4-
3',11234567,'payno@gmail.com','Bangalore',560068,'Marthahalli');

INSERT INTO FERRY_INFO(Ferry_id,Ferry_name, Seat_availability, source , dest ,
Vehicle_spots, Type_of_ferry) VALUES
('A102','Black pearl',90,'Australia','England',85,'Deluxe'),
('B102','Scavenger',100,'Australia','Japan',50,'Regular'),
('G2222','Avenger',28,'India','Germany',12,'Luxury'),
('A1021D','Titanic',70,'China','England',10,'Cargo'),
('A122B','Black swan',45,'Russia','Canada',45,'Deluxe'),
('PE502','Bisleri',91,'Russia','Netherlands',60,'Regular');

INSERT INTO TICKET(Ticket_no,Ferry_name, Ferry_id, Seat_no, Admin_id
,Passenger_name) VALUES
(124,'Black pearl','A102','H12',2,'Jack'),
(1253,'Scavenger','B102','E100',5,'Quill'),
(756,'Avenger','G2222','A28',3,'Yasmin'),
(1097,'Titanic','A1021D','T56',6,'Chadeler'),
(12,'Black swan','A122B','J90',7,'Rachel'),
(645,'Bisleri','PE502','P23',2,'Joey');

INSERT INTO PAYMENT(Trans_id, Amount ,Date_trans ,
Mode_of_payment,Ticket_no) VALUES
(1000919,1500,'2021-09-11','UPI',124),
(2000283,1000,'2020-12-8','CASH',1253),
(45355526,5000,'2020-09-7','CASH',756),
(563637464,2567,'2019-05-26','NET BANKING',1097);

INSERT INTO PASSENGER(P_id,Fname, Mname ,Lnamme, Phone_no, Address,
```

```
Email, Admin_id) VALUES
(334,'Cassie','Williams','Horan',16274533,'No.5, AECS ,
Bangalore','cwh54@yahoo.co.in',5),
(400,'Rue','Williams','Horan',16274533,'No.5, AECS ,
Bangalore','rwh9@yahoo.co.in',5),
(064,'Monica','Malvi','Geller',273547,'No.3, Neeladiri ,
Chennai','cwh54@yahoo.co.in',7),
(001,'Arjun','Venkata','Das',8991727,'No.82,Twin towers ,
Mumbai','cwh54@yahoo.co.in',2),
(093,'Rahul','Jagadesh','Annamalai',91754726,'No.109D, Shriram Spurthi ,
Delhi','cwh54@yahoo.co.in',3),
(982,'Ninad','James','Swift',9097436,'No.K12, UB City ,
Chennai','cwh54@yahoo.co.in',4);

INSERT INTO ROUTE_INFO(Source, Dest , Travel_time , Ferry_id) VALUES
('Mumbai','Boston',48,'A102'),
('Hong kong','Sydney',16,'A1021D'),
('Mumbai','Seattle',24,'B102'),
('Yeman','New york',15,'G2222'),
('Mumbai','London',18,'PE502');

INSERT INTO FERRY_STAFF(Captain, Vice_captain, Helpers, Ferry_id ) VALUES
('Jagadish','Kumar',15,'A102'),
('Jacobs','Nate',10,'A1021D'),
('Gunther','Vijay',10,'B102'),
('Jyoti','Surya',5,'G2222'),
('Arjun','Das',25,'PE502');

INSERT INTO VEHICLE_SLOT(Slot_no, Location , Ticket_no ) VALUES
(10,'Upper deck',1253),
(24,'Upper deck',756),
(19,'Patio',12),
(4,'Lower deck',645);

INSERT INTO ACCOMPANY(Id, P_id) VALUES
(1093,334),
(12,064),
(872,001);
```

==========

On executing select statements, all the rows from the tables get displayed.

#	Time	Action	Message	Duration / Fetch
1	17:55:29	SELECT * FROM ADMIN LIMIT 0, 1000	7 row(s) returned	0.015 sec / 0.000 sec
2	17:55:29	SELECT * FROM FERRY_INFO LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
3	17:55:29	SELECT * FROM TICKET LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
4	17:55:29	SELECT * FROM PAYMENT LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
5	17:55:29	SELECT * FROM PASSENGER LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
6	17:55:29	SELECT * FROM ROUTE_INFO LIMIT 0, 1000	5 row(s) returned	0.015 sec / 0.000 sec

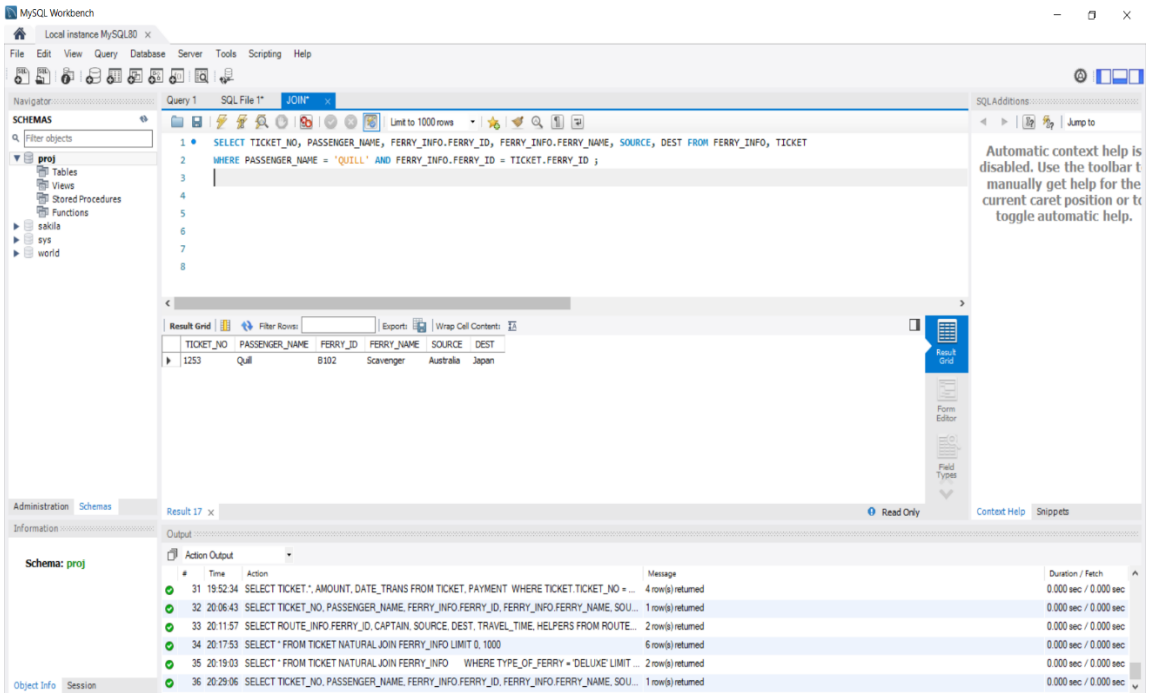
#	Time	Action	Message	Duration / Fetch
5	17:55:29	SELECT * FROM PASSENGER LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
6	17:55:29	SELECT * FROM ROUTE_INFO LIMIT 0, 1000	5 row(s) returned	0.015 sec / 0.000 sec
7	17:55:29	SELECT * FROM FERRY_STAFF LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
8	17:55:29	SELECT * FROM ROUTE_INFO LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
9	17:55:29	SELECT * FROM ACCOMPANY LIMIT 0, 1000	3 row(s) returned	0.016 sec / 0.000 sec

6. Join Queries

1. TO DISPLAY THE TICKET NO, FERRY NAME, FERRY ID, SOUCE AND THE DESTINATION OF THE FERRY FOR THE PASSENGER “QUILL”.

SQL QUERY :

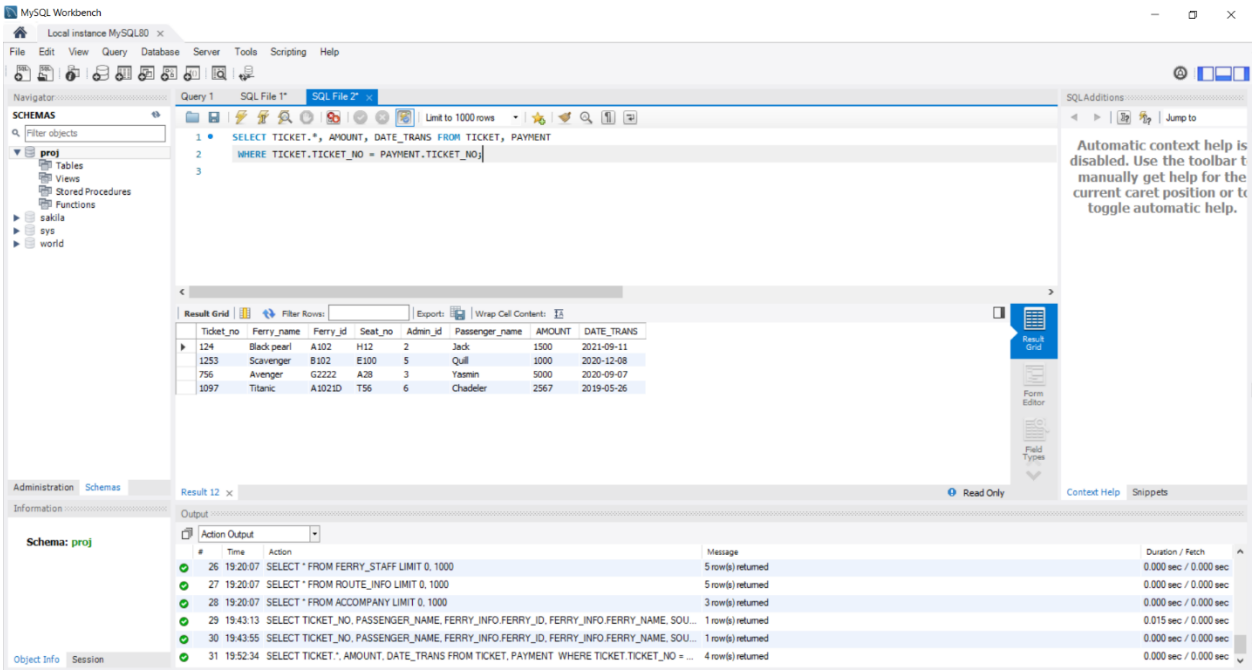
```
SELECT TICKET_NO, PASSENGER_NAME, FERRY_INFO.FERRY_ID,
    FERRY_INFO.FERRY_NAME, SOURCE, DEST FROM FERRY_INFO,
    TICKET
WHERE PASSENGER_NAME = 'QUILL' AND
FERRY_INFO.FERRY_ID = TICKET.FERRY_ID ;
```



2. TO DISPLAY ALL THE TICKET DETAILS ALONG WITH THE TICKET AMOUNT AND DATA OF TRANSACTION.

SQL QUERY :

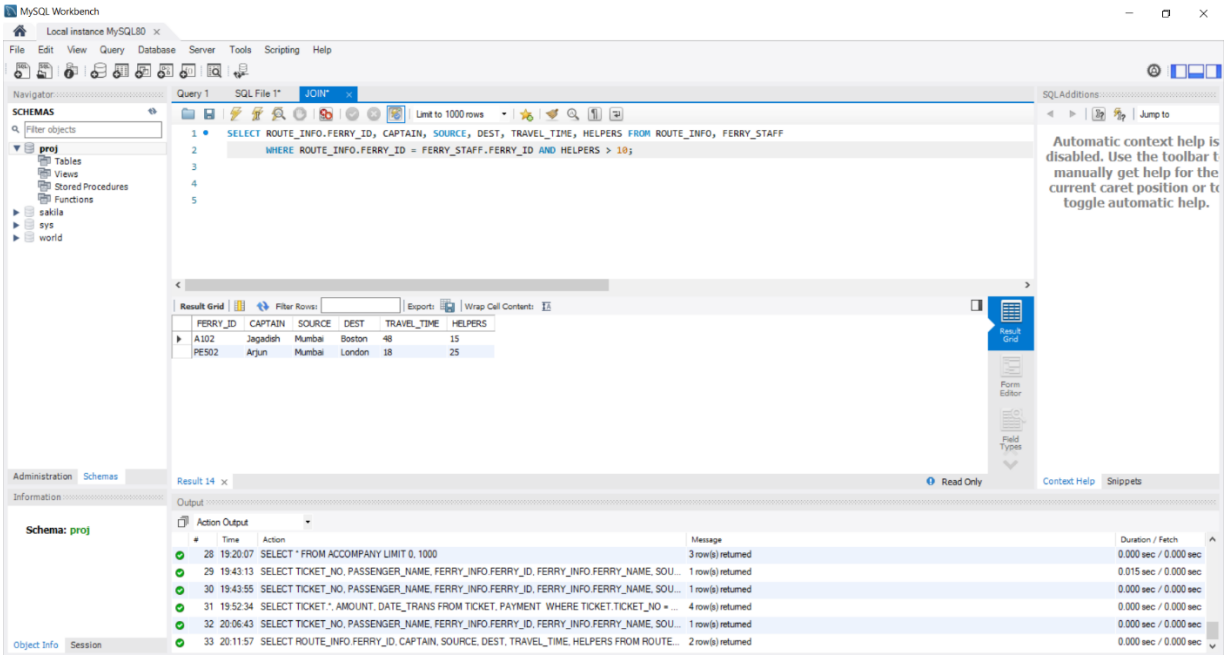
```
SELECT TICKET.*, AMOUNT, DATE_TRANS FROM TICKET, PAYMENT
    WHERE TICKET.TICKET_NO = PAYMENT.TICKET_NO;
```



3. TO DISPLAY THE ROUTE INFORMATION OF THE FERRIES WITH THE CAPTAIN NAME WHICH HAS MORE THAN 10 HELPERS.

SQL QUERY :

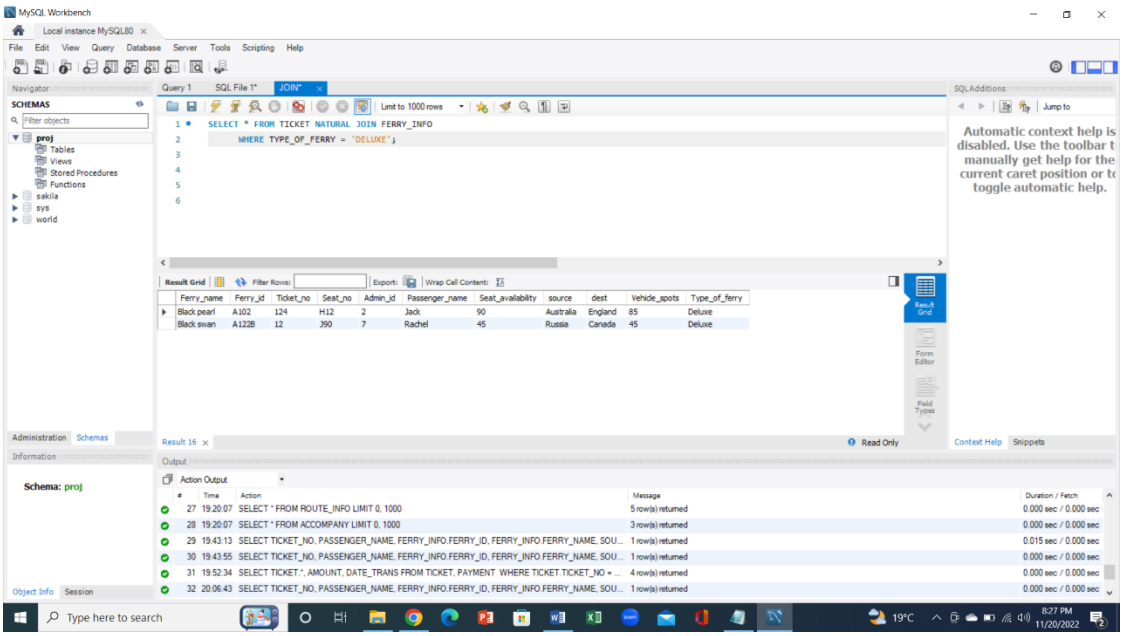
```
SELECT ROUTE_INFO.FERRY_ID, CAPTAIN, SOURCE, DEST,
    TRAVEL_TIME, HELPERS FROM ROUTE_INFO, FERRY_STAFF
    WHERE ROUTE_INFO.FERRY_ID = FERRY_STAFF.FERRY_ID AND
    HELPERS > 10;
```



4. DISPLAYING ALL THE DETAILS OF TICKET AND FERRY INFORMATION FOR DELUXE TYPE FERRY USING NATURAL JOIN.

SQL QUERY :

SELECT * FROM TICKET NATURAL JOIN FERRY_INFO
WHERE TYPE_OF_FERRY = 'DELUXE' ;

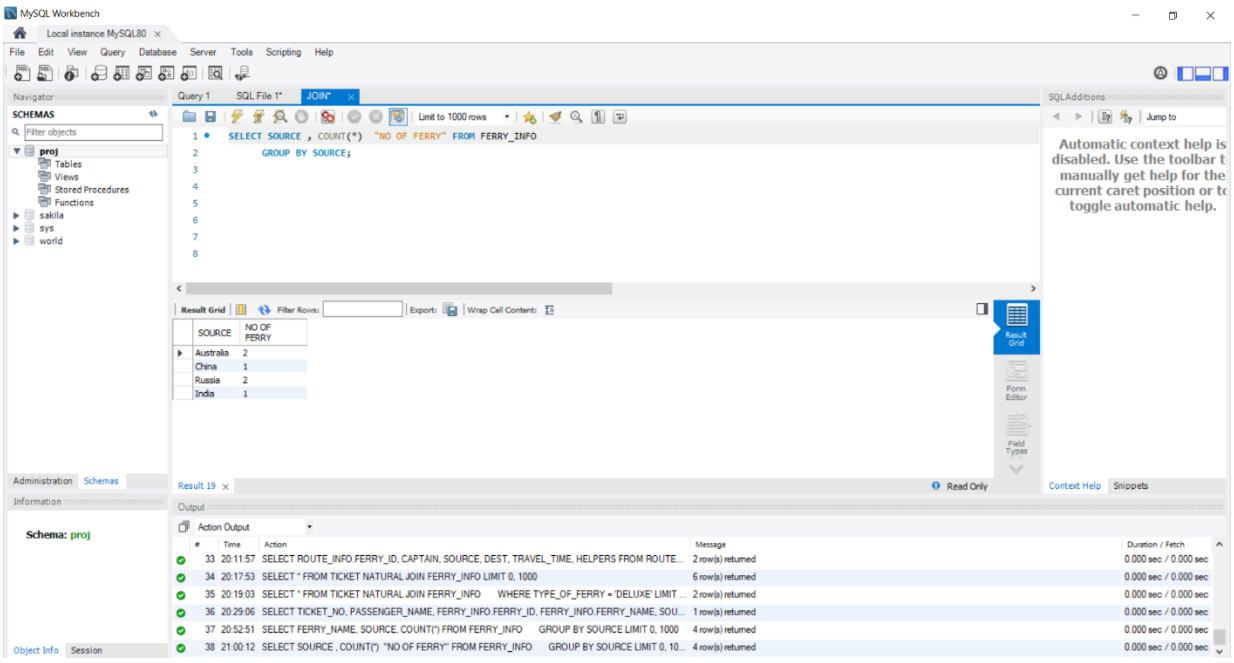


7. Aggregate Functions

1. TO DISPLAY THE NUMBER OF FERRIES FROM EACH STARTING (SOURCE) PLACE.

SQL QUERY :

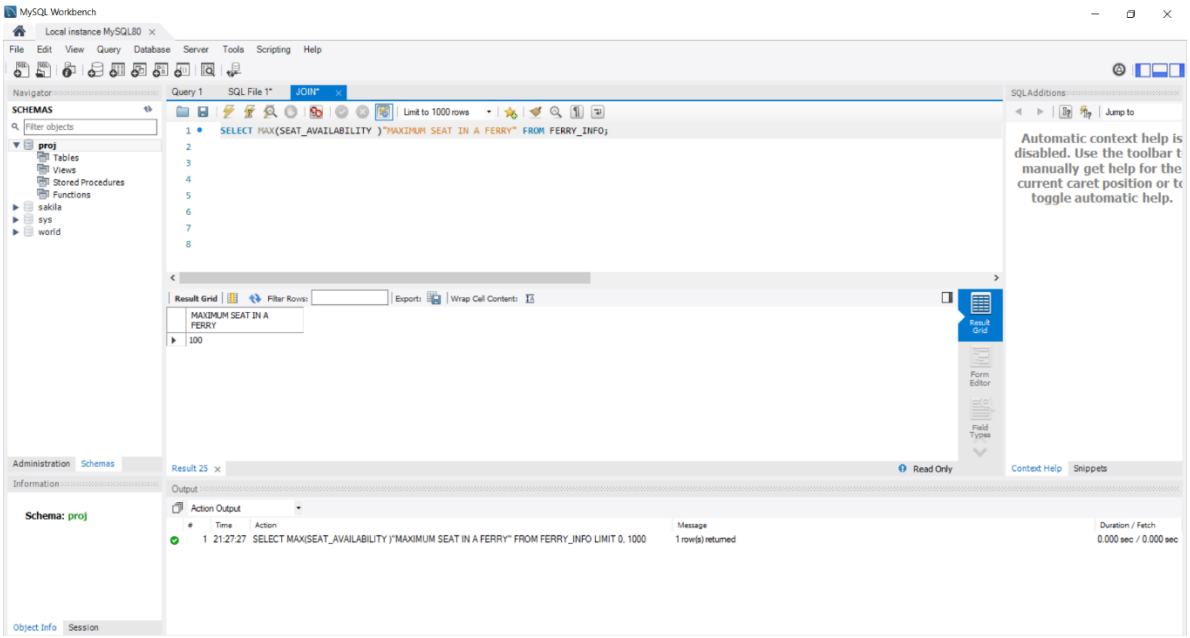
```
SELECT SOURCE , COUNT(*)  "NO OF FERRY" FROM FERRY_INFO
    GROUP BY SOURCE;
```



2. TO DISPLAY THE MAXIMUM SEAT AVAILABILITY IN A FERRY.

SQL QUERY :

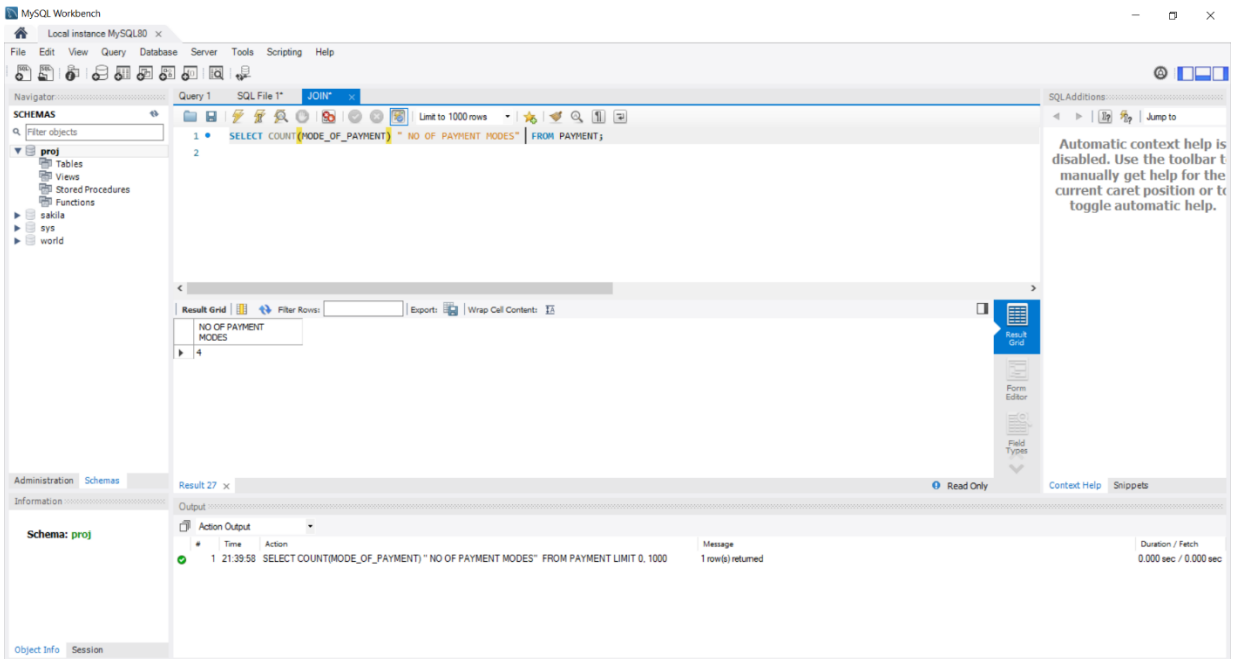
```
SELECT MAX(SEAT_AVAILABILITY )"MAXIMUM SEAT IN A FERRY"
    FROM FERRY_INFO;
```



3. TO DISPLAY THE DIFFERENT NUMBER OF PAYMENT MODES.

SQL QUERY :

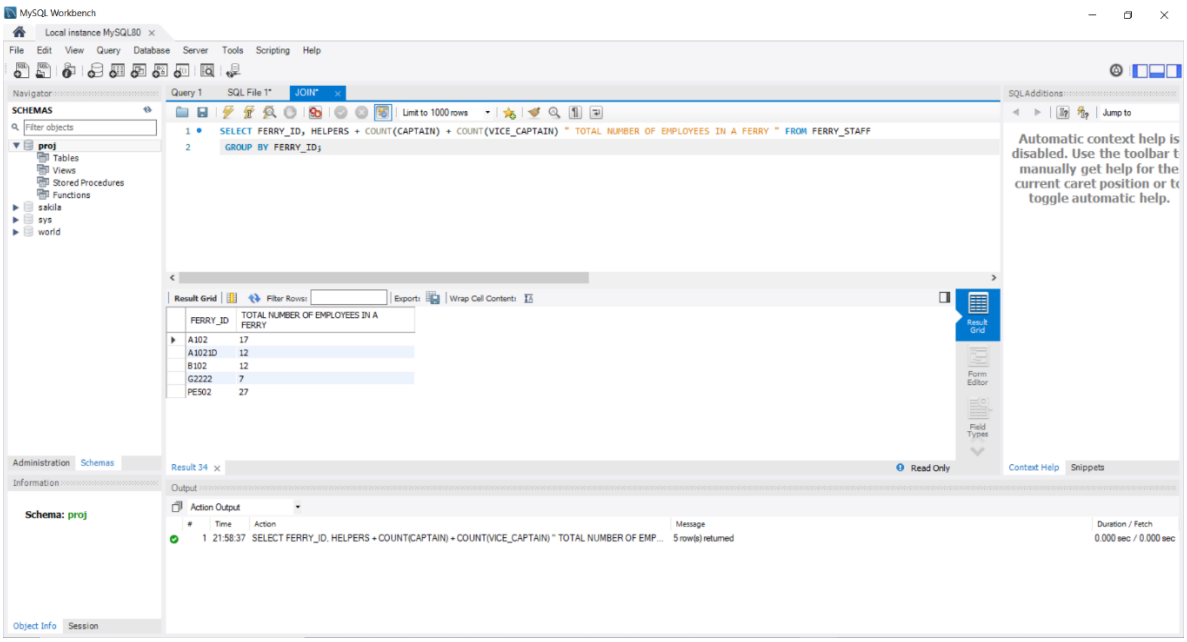
SELECT COUNT(MODE_OF_PAYMENT) " NO OF PAYMENT MODES"
FROM PAYMENT;



4. TO DISPLAY THE TOTAL NUMBER OF STAFF IN A FERRIES.

SQL QUERY :

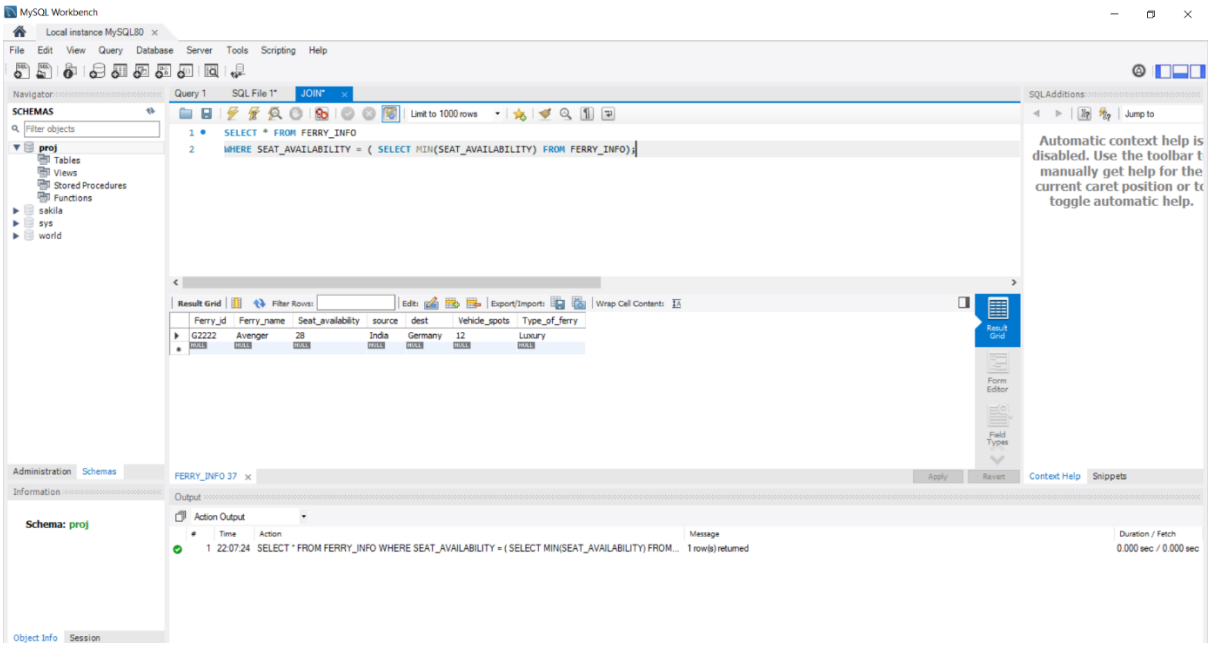
```
SELECT FERRY_ID, HELPERS + COUNT(CAPTAIN) +
    COUNT(VICE_CAPTAIN) " TOTAL NUMBER OF EMPLOYEES IN A
    FERRY " FROM FERRY_STAFF
GROUP BY FERRY_ID;
```



5. TO DISPLAY THE FERRY DETAILS WHICH HAS MINIMUM NUMBER OF SEAT AVAILABILITY.

SQL QUERY :

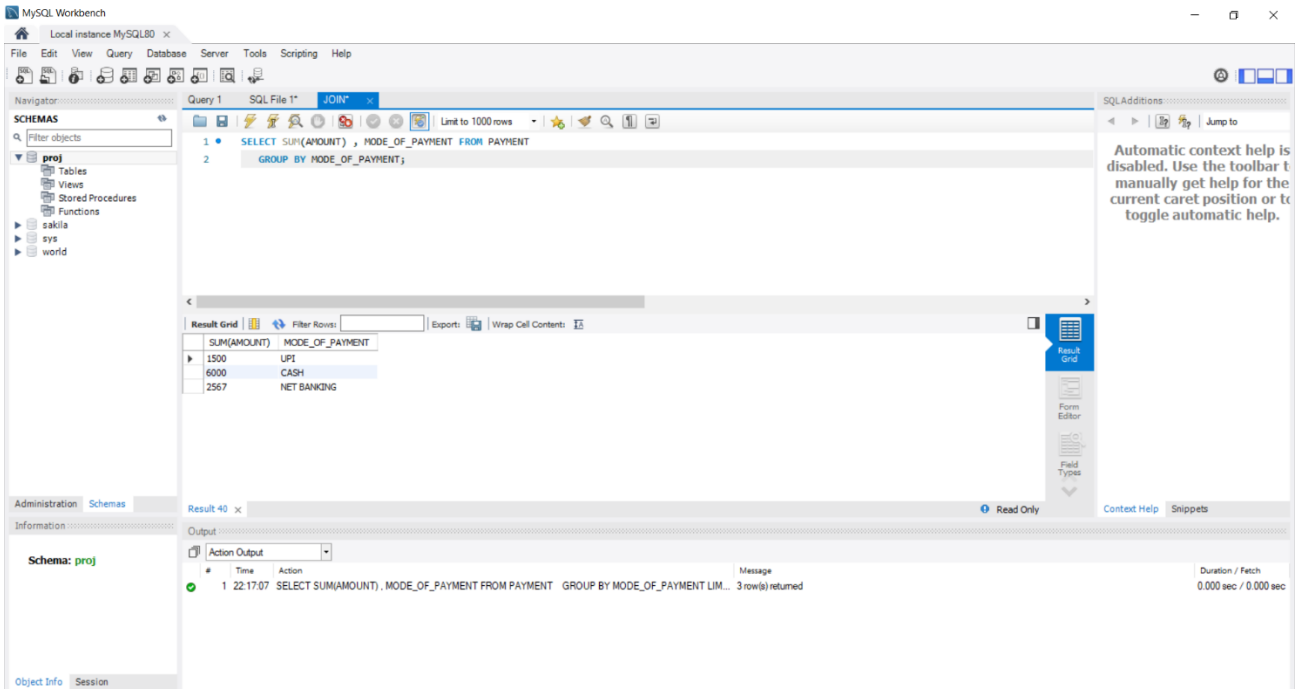
```
SELECT * FROM FERRY_INFO
WHERE SEAT_AVAILABILITY = ( SELECT MIN(SEAT_AVAILABILITY)
    FROM FERRY_INFO);
```



6. DISPLAY THE TOTAL NUMBER OF AMOUNT RECEIVED IN EACH MODE OF PAYMENT.

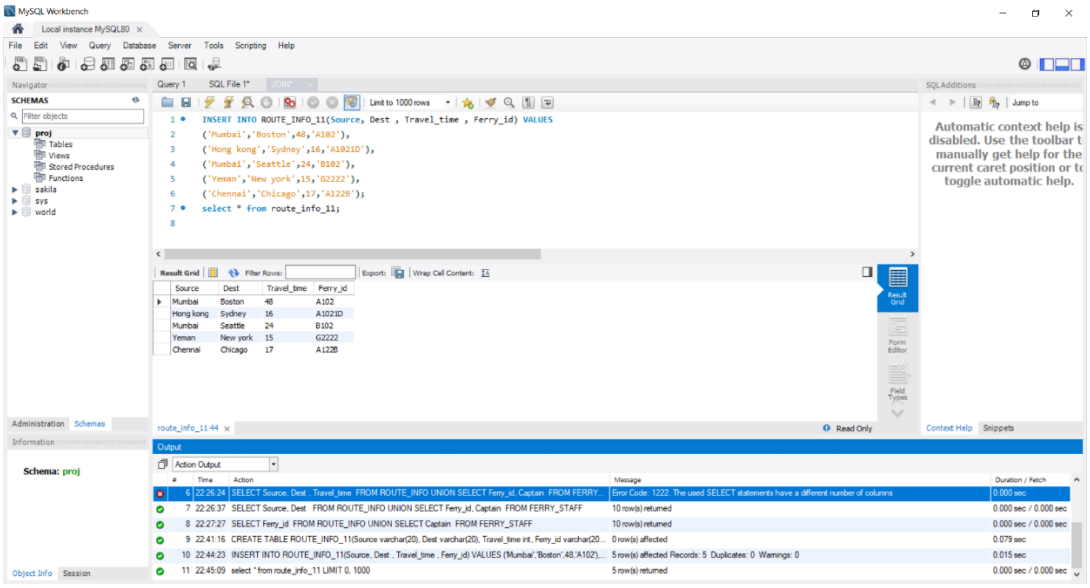
SQL QUERY :

```
SELECT SUM(AMOUNT) , MODE_OF_PAYMENT FROM PAYMENT
  GROUP BY MODE_OF_PAYMENT;
```

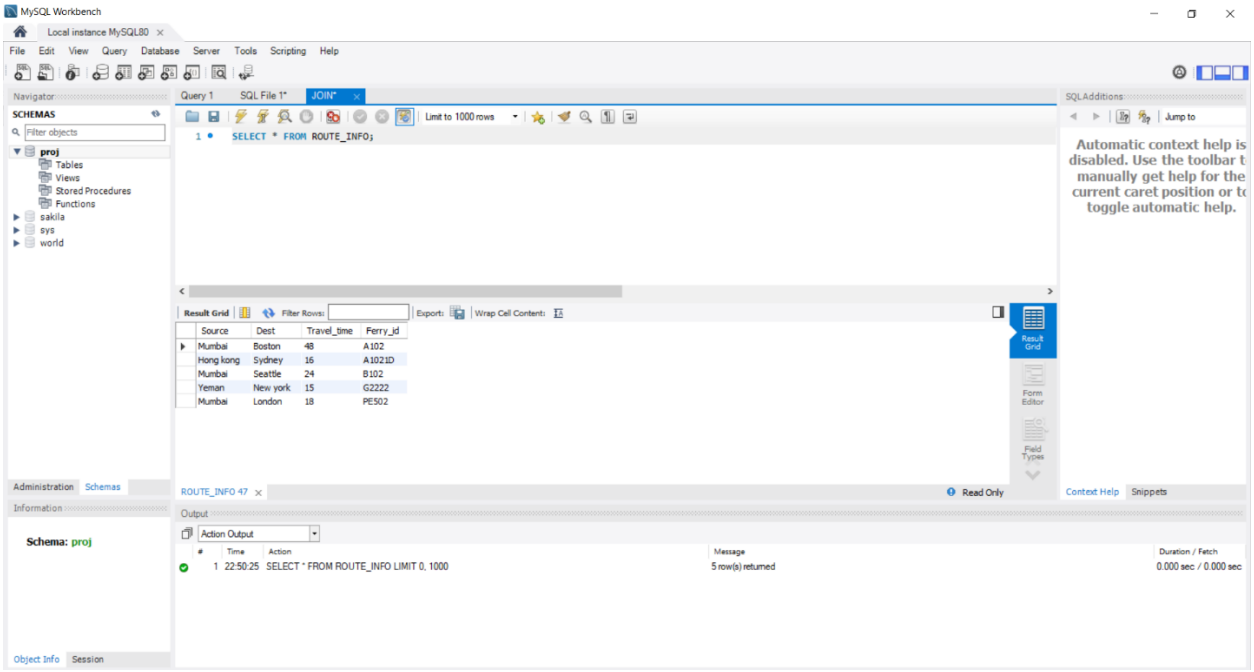


8. Set Operations

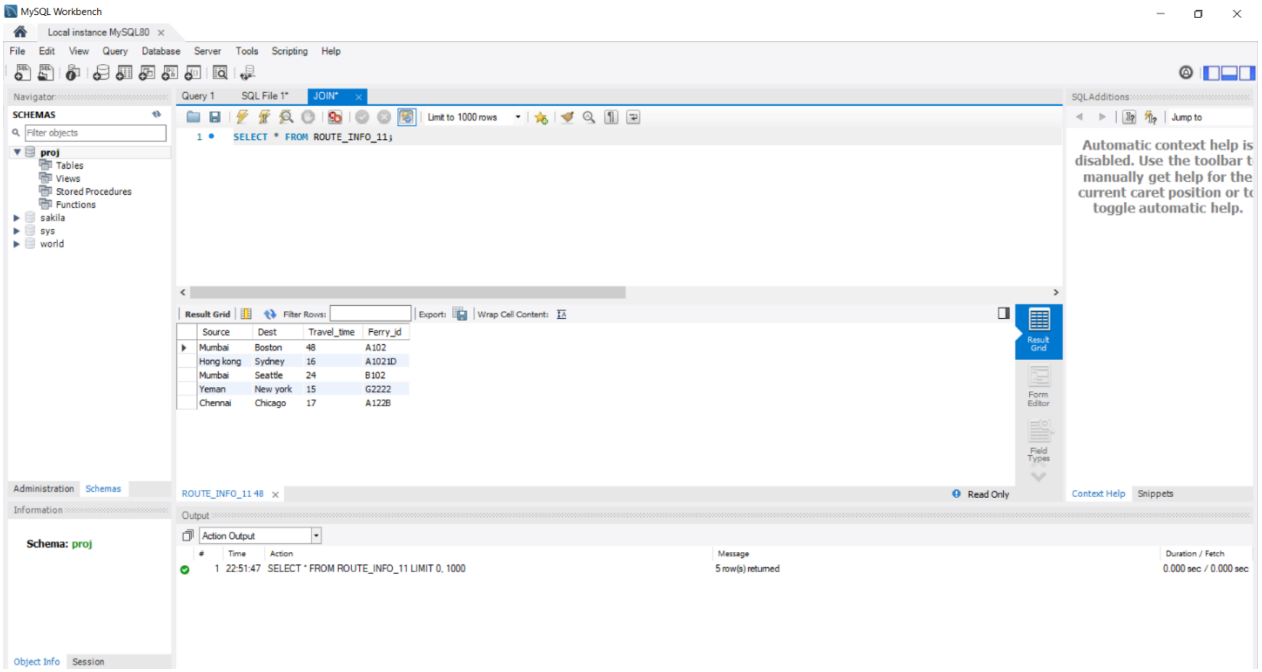
NEW TABLE *ROUTE_INFO_11* FOR SET OPERATIONS:



ROUTE_INFO TABLE:



ROUTE_INFO_11 TABLE:

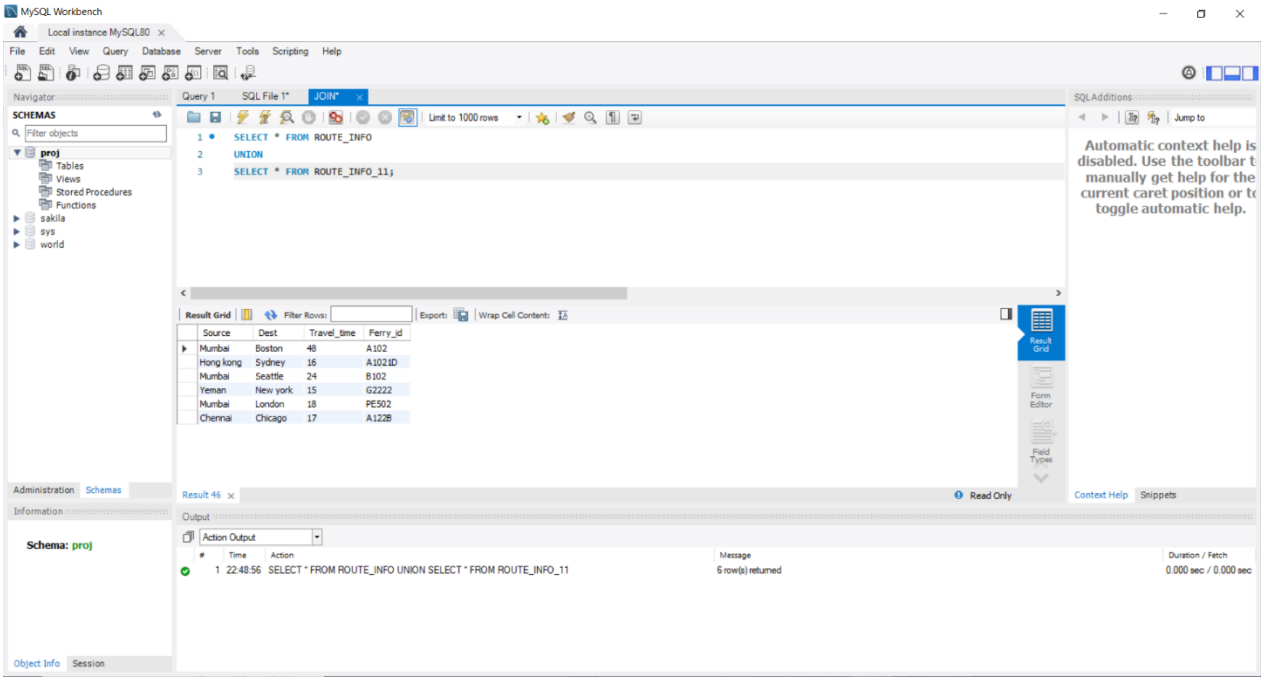


1. UNION OPERATION :

To perform union between the tables ROUTE_INFO and ROUTE_INFO_11.

SQL QUERY :

```
SELECT * FROM ROUTE_INFO
UNION
SELECT * FROM ROUTE_INFO_11;
```



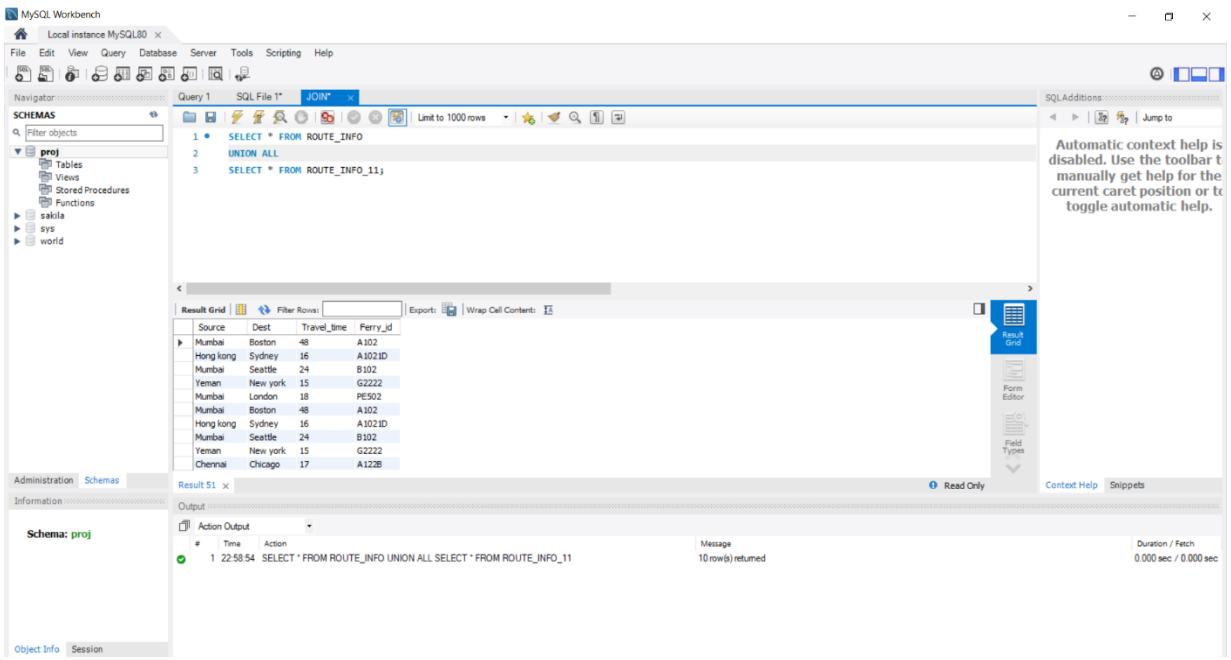
2. UNION ALL:

To perform union all operation between the table ROUTE_INFO and ROUTE_INFO_11.

SQL QUERY :

```
SELECT * FROM ROUTE_INFO
UNION ALL
SELECT * FROM ROUTE_INFO_11;
```

NOTE : Duplicate rows will be not be eliminated

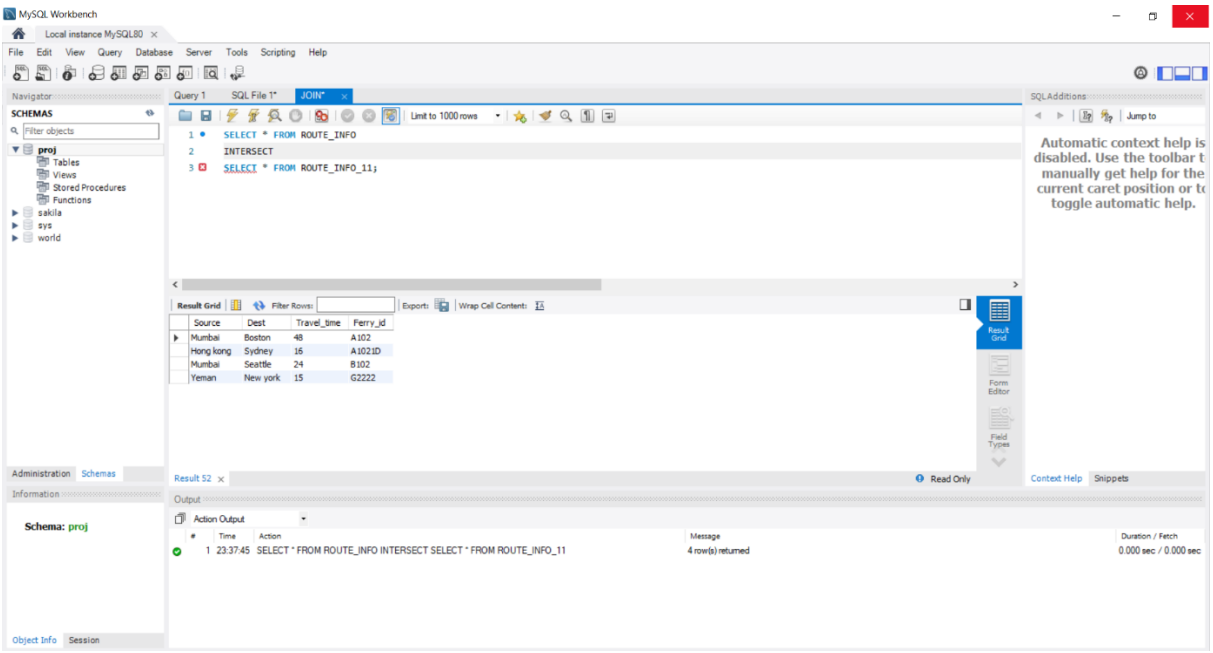


3. INTERSECT:

To perform intersect operation between the table ROUTE_INFO and ROUTE_INFO_11.

SQL QUERY :

```
SELECT * FROM ROUTE_INFO
INTERSECT
SELECT * FROM ROUTE_INFO_11;
```



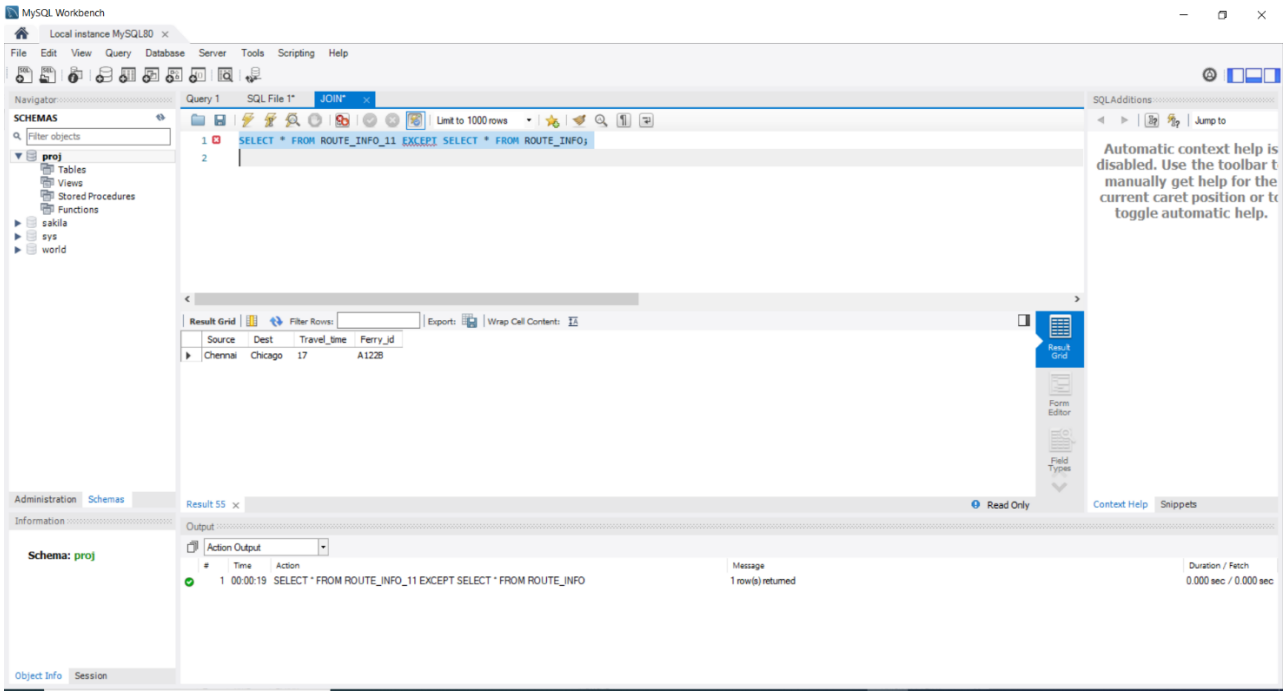
4. EXCEPT :

To perform minus operation between the table ROUTE_INFO_11 and ROUTE_INFO.

That is, to display the details present in ROUTE_INFO _11 table but not in ROUTE_INFO table.

SQL QUERY :

```
SELECT * FROM ROUTE_INFO_11 EXCEPT SELECT * FROM ROUTE_INFO;
```


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
proj
Tables
Views
Stored Procedures
Functions
sakila
sys
world
Query 1
SQL File 1*
JOIN*
Limit to 1000 rows
1 SELECT * FROM ROUTE_INFO_11 EXCEPT SELECT * FROM ROUTE_INFO;
2
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or to toggle automatic help.
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Source Dest Travel_time Ferry_id
Chennai Chicago 17 A122B
Result Grid
Form Editor
Field Types
Administration Schemas
Information
Schema: proj
Result 55
Read Only
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 00:00:19 SELECT * FROM ROUTE_INFO_11 EXCEPT SELECT * FROM ROUTE_INFO 1 row(s) returned 0.000 sec / 0.000 sec
Object Info Session

9. Functions and Procedures

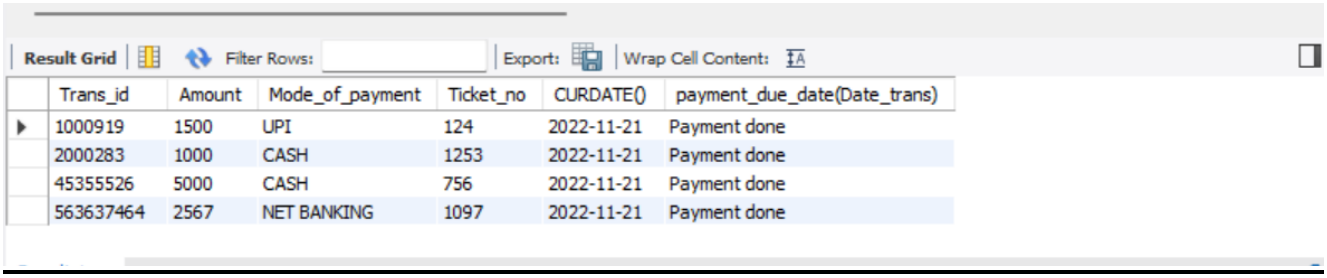
FUNCTION:

Create a function to check whether the payment for the ticket has been completed or not.

SQL QUERY:

```
DELIMITER $$
CREATE FUNCTION payment_due_date(payment_date DATE)
RETURNS VARCHAR(50)
 BEGIN
   DECLARE pay VARCHAR(50);
   IF CURRENT_DATE() > payment_date THEN
          SET pay = 'Payment done';
   ELSEIF CURRENT_DATE() <= payment_date THEN
          SET pay = 'Payment not done';
   END IF;
       RETURN pay;
END; $$
DELIMITER ;

SELECT
Trans_id,Amount,Mode_of_payment,Ticket_no,CURDATE(),payment_due_date(Date
_trans)  FROM  PAYMENT;
```

OUTPUT:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Trans_id	Amount	Mode_of_payment	Ticket_no	CURDATE()	payment_due_date(Date_trans)
1000919	1500	UPI	124	2022-11-21	Payment done
2000283	1000	CASH	1253	2022-11-21	Payment done
45355526	5000	CASH	756	2022-11-21	Payment done
563637464	2567	NET BANKING	1097	2022-11-21	Payment done

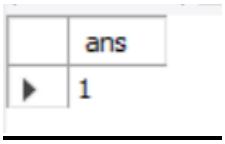
PROCEDURE:

Create a procedure to check which Admin_id corresponds to which Passenger id.

SQL QUERY:

```
DELIMITER $$
CREATE procedure user_booking(
IN UID int, OUT msg varchar(30))
BEGIN
DECLARE count_tickets int;
set count_tickets= (SELECT Ticket_no from TICKET where Admin_id=UID);

IF count_tickets = 0 THEN
  set msg= 'Booked zero tickets';

ELSE
  set msg=(SELECT P_id FROM PASSENGER WHERE Admin_id = UID);

END IF;

END;$$
DELIMITER ;

SET @M="";
CALL user_booking(2,@M);
SELECT @M as ans;
```

OUTPUT:



	ans
▶	1

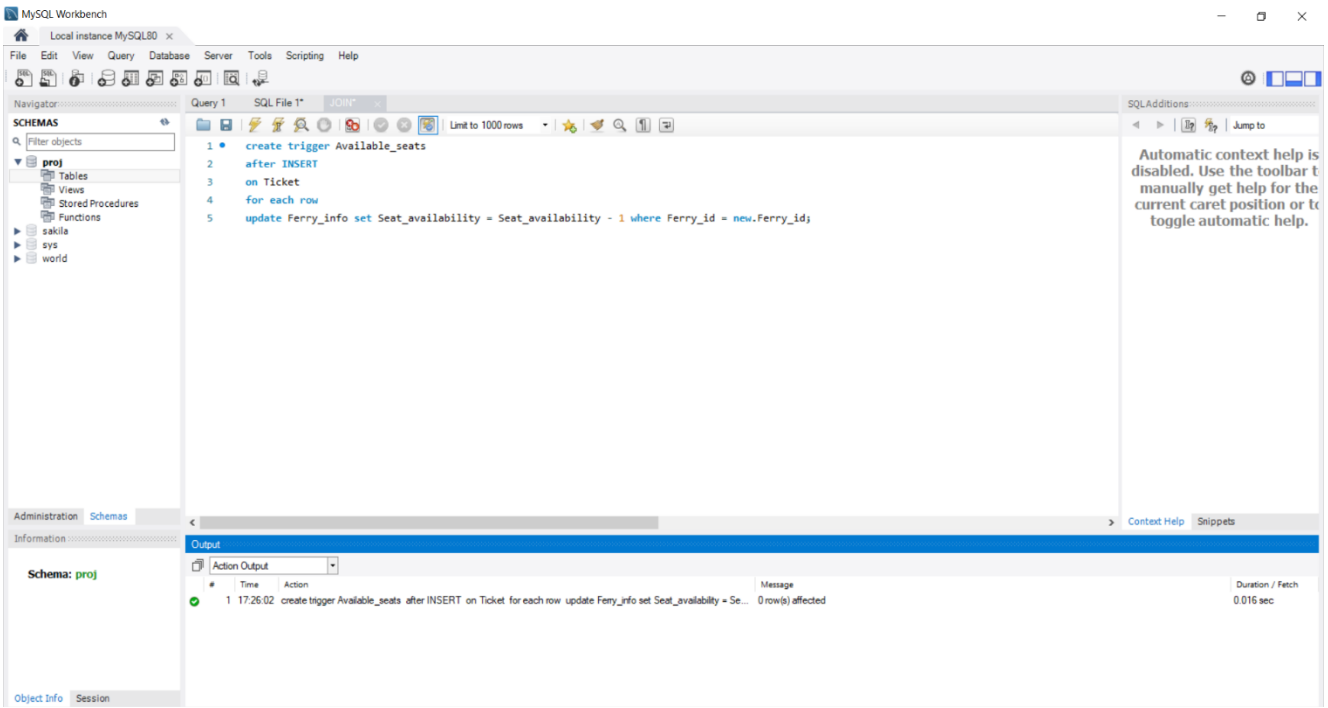
10. Triggers and Cursors

TRIGGER:

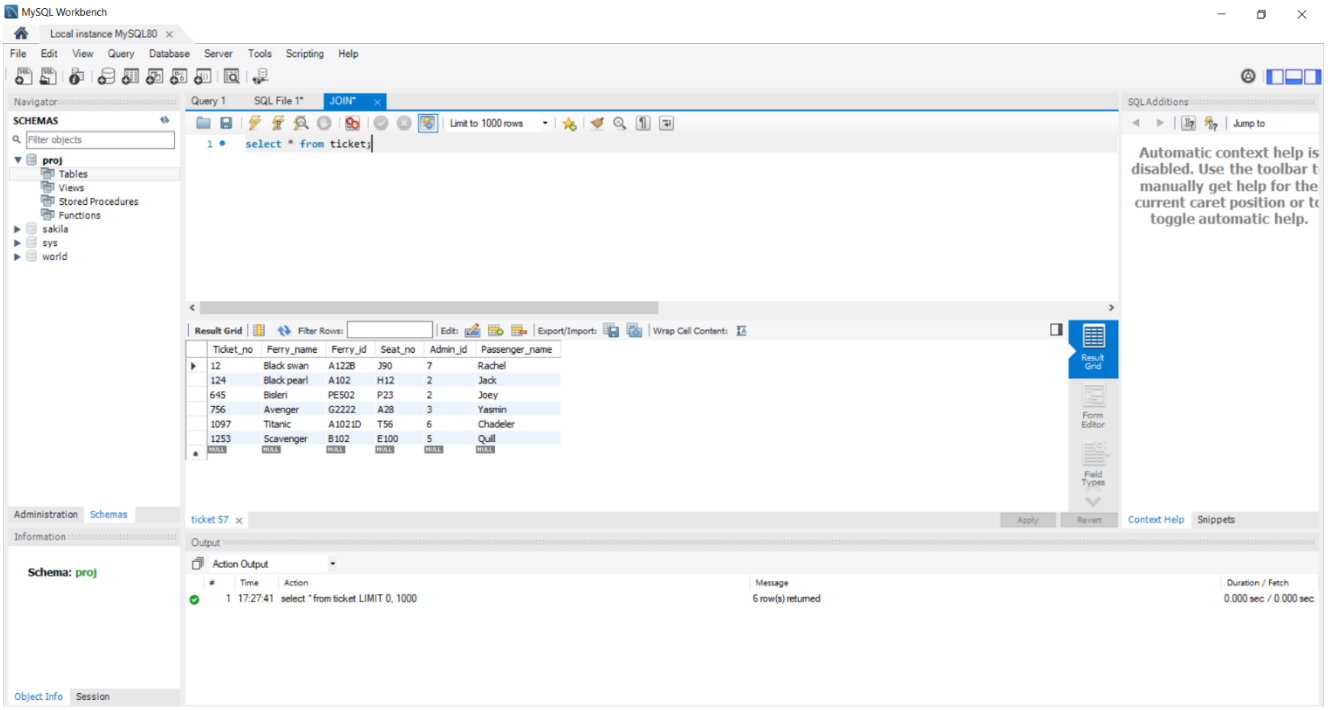
When you inserts the data into the TICKET database a trigger should automatically invoke and decrements the Seat_availability attribute in FERRY_INFO database by 1 so that a proper track of No of available seats can be maintained.

SQL QUERY:

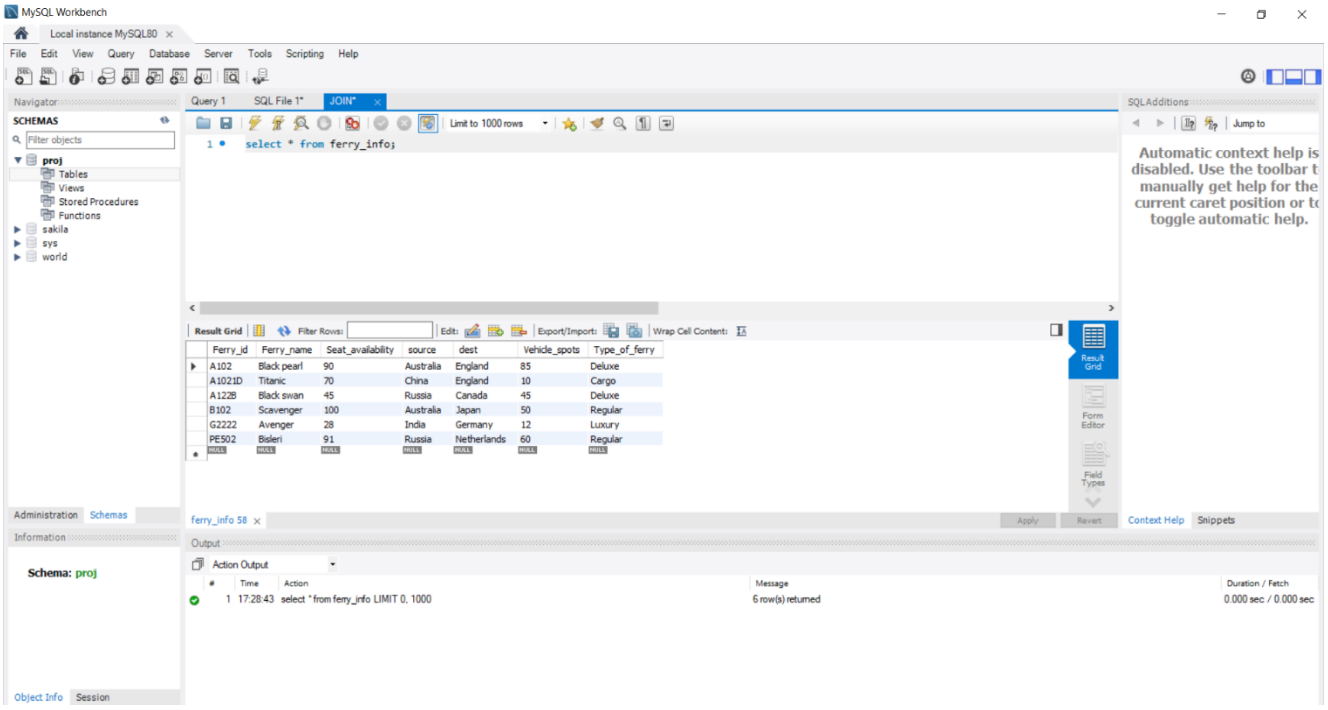
```
create trigger Available_seats
after INSERT
on Ticket
for each row
update Ferry_info set Seat_availability = Seat_availability - 1 where Ferry_id =
    new.Ferry_id;
```



TICKET TABLE :

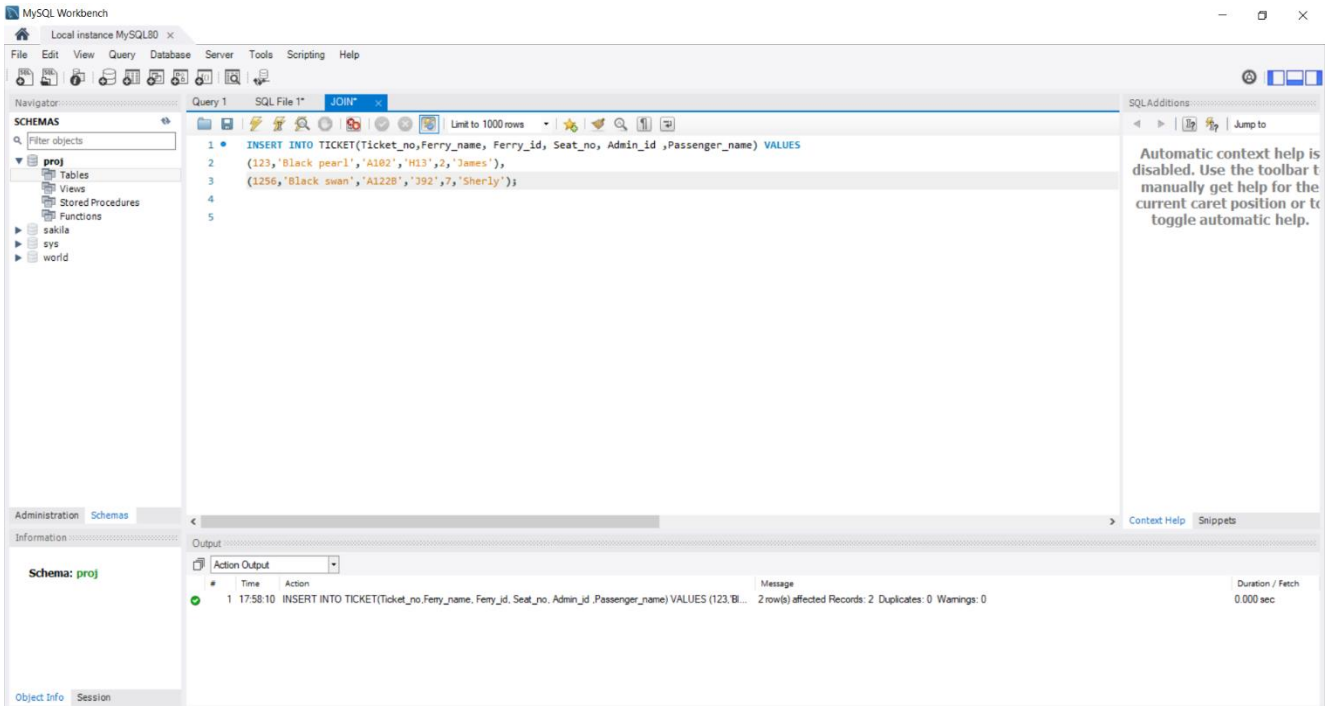


FERRY_INFO TABLE :



INSERTING 2 RECORDS IN TICKET DATABASE:

```
INSERT INTO TICKET(Ticket_no,Ferry_name, Ferry_id, Seat_no, Admin_id
    ,Passenger_name) VALUES
(123,'Black pearl','A102','H13',2,'James'),
(1256,'Black swan','A122B','J92',7,'Sherly');
```



11. Developing a Frontend

CODE FOR FRONTEND:

```
import mysql.connector
from tkinter import *
from tkinter import messagebox
from tkinter import ttk
import random
import time
import datetime


class ferry:
  def __init__(self,root):
    self.root=root
    self.root.title("FERRY MANAGEMENT SYSTEM")
    self.root.geometry("1400x500+0+0")

    self.FerryId = StringVar()
    self.Ferryname=StringVar()
    self.Seat = IntVar()
    self.From = StringVar()
    self.To = StringVar()
    self.Type = StringVar()
    self.Slot = IntVar()


    lbltitle=Label(self.root,bd=20,relief=RIDGE,text="FERRY MANAGEMENT
SYSTEM",fg="blue",bg="white",font=("times new roman",50,"bold"))
    lbltitle.pack(side=TOP,fill=X)

    ##DATAFRAME

    Dataframe=Frame(self.root,bd=20,relief=RIDGE)
    Dataframe.place(x=0,y=130,width=1400,height=250)

    DataframeLeft=LabelFrame(Dataframe,bd=10,padx=10,relief=RIDGE,
                        font=("times new roman",12,"bold"),text="Ferry Infromation")

    DataframeLeft.place(x=0,y=5,width=800,height=200)
```

```
    DataframeRight=LabelFrame(Dataframe,bd=10,padx=10,relief=RIDGE,
                              font=("times new roman",12,"bold"),text="Route Infromation")

    DataframeRight.place(x=810,y=5,width=500,height=200)

    ##BUTTON FRAME

    Buttonframe=Frame(self.root,bd=20,relief=RIDGE)
    Buttonframe.place(x=0,y=400,width=1400,height=70)

    ##DETAILS FRAME

    Detailsframe=Frame(self.root,bd=20,relief=RIDGE)
    Detailsframe.place(x=0,y=480,width=1400,height=180)

    ##DATA FRAME LEFT

    lblferryid = Label(DataframeLeft,text="Ferry ID",font=("times new
roman",12,"bold"),padx=2,pady=6)
    lblferryid.grid(row=0,column=0,sticky=W)
    txt1 = Entry(DataframeLeft, textvariable=self.FerryId,font=("times new roman", 12,
"bold"), width=25)
    txt1.grid(row=0, column=1)

    lblfname=Label(DataframeLeft,font=("times new roman",12,"bold"),text="Ferry
Name",padx=2,pady=6)
    lblfname.grid(row=1,column=0,sticky=W)
    txt2=Entry(DataframeLeft,textvariable=self.Ferryname,font=("times new
roman",12,"bold"),width=25)
    txt2.grid(row=1,column=1)

    lblseat = Label(DataframeLeft, font=("times new roman", 12, "bold"), text="Seat
Availability", padx=2, pady=6)
    lblseat.grid(row=2, column=0, sticky=W)
    txt3 = Entry(DataframeLeft, textvariable=self.Seat, font=("times new roman", 12,
"bold"), width=25)
    txt3.grid(row=2, column=1)

    lblfrom = Label(DataframeLeft, font=("times new roman", 12, "bold"), text="From",
padx=2, pady=6)
    lblfrom.grid(row=3, column=0, sticky=W)
    txt4 = Entry(DataframeLeft, textvariable=self.From, font=("times new roman", 12,
```

```
"bold"), width=25)
    txt4.grid(row=3, column=1)

    lblto = Label(DataframeLeft, font=("times new roman", 12, "bold"), text="To",
padx=2, pady=6)
    lblto.grid(row=4, column=0, sticky=W)
    txt5 = Entry(DataframeLeft, textvariable=self.To, font=("times new roman", 12,
"bold"), width=25)
    txt5.grid(row=4, column=1)

    lbltype = Label(DataframeLeft, font=("times new roman", 12, "bold"), text="Type",
padx=2, pady=6)
    lbltype.grid(row=0, column=30, sticky=W)
    txt6 = Entry(DataframeLeft, textvariable=self.Type, font=("times new roman", 12,
"bold"), width=25)
    txt6.grid(row=0, column=31)

    lblslot = Label(DataframeLeft, font=("times new roman", 12, "bold"), text="Vehicle
Slots", padx=2, pady=6)
    lblslot.grid(row=1, column=30, sticky=W)
    txt7 = Entry(DataframeLeft, textvariable=self.Slot, font=("times new roman", 12,
"bold"), width=25)
    txt7.grid(row=1, column=31)



    ##DATAFRAME RIGHT

    self.txtInformation=Text(DataframeRight,font=("times new
roman",12,"bold"),width=50,padx=2,pady=6)
    self.txtInformation.grid(row=0,column=0)

    ##BUTTONS

    btninfo=Button(Buttonframe,text="Display
details",command=self.display,bg="green",fg="white",font=("times new
roman",12,"bold"),width=15,height=1,padx=2,pady=6)
    btninfo.grid(row=0,column=0)

    btninsert = Button(Buttonframe, text="Insert",command=self.insert, bg="green",
fg="white", font=("times new roman", 12, "bold"),width=15, height=1, padx=2,
pady=6)
    btninsert.grid(row=0, column=2)
```

```
    btnupdate = Button(Buttonframe, text="Update",command=self.update, bg="green",
fg="white", font=("times new roman", 12, "bold"),width=15, height=1, padx=2,
pady=6)
    btnupdate.grid(row=0, column=4)

    btnroute = Button(Buttonframe,text="Route Info",command=self.information,
bg="green", fg="white", font=("times new roman", 12, "bold"),width=15, height=1,
padx=2, pady=6)
    btnroute.grid(row=0, column=6)

    btndelete = Button(Buttonframe, text="Delete", command=self.delete, bg="green",
fg="white",
                   font=("times new roman", 12, "bold"), width=15, height=1, padx=2,
pady=6)
    btndelete.grid(row=0, column=8)

    btnclear = Button(Buttonframe, text="Clear", command=self.clear, bg="green",
fg="white",
                   font=("times new roman", 12, "bold"), width=15, height=1, padx=2,
pady=6)
    btnclear.grid(row=0, column=10)

    btnexit = Button(Buttonframe, text="Exit", command=self.exit, bg="green",
fg="white",
                   font=("times new roman", 12, "bold"), width=15, height=1, padx=2,
pady=6)
    btnexit.grid(row=0, column=12)

    ##TABLE
    ##SCROLLBAR
    scroll_x=Scrollbar(Detailsframe,orient=HORIZONTAL)
    scroll_y = Scrollbar(Detailsframe,orient=VERTICAL)

self.ferry_table=ttk.Treeview(Detailsframe,column=("FerryID","FerryName","SeatAvai
lability","From","To","VehicleSpots","Type"),xscrollcommand=scroll_x.set,yscrollcom
mand=scroll_y.set)

    scroll_x.pack(side=BOTTOM,fill=X)
    scroll_y.pack(side=RIGHT,fill=Y)

    scroll_x=ttk.Scrollbar(command=self.ferry_table.xview)
    scroll_y = ttk.Scrollbar(command=self.ferry_table.yview)
```

```
    self.ferry_table.heading("FerryID", text="Ferry ID")
    self.ferry_table.heading("FerryName",text="Ferry name")
    self.ferry_table.heading("SeatAvailability", text="Seat availability")
    self.ferry_table.heading("From", text="From")
    self.ferry_table.heading("To", text="To")
    self.ferry_table.heading("VehicleSpots", text="Vehicle spots")
    self.ferry_table.heading("Type", text="Type")


    self.ferry_table["show"]="headings"

    self.ferry_table.column("FerryID", width=60)
    self.ferry_table.column("FerryName", width=60)
    self.ferry_table.column("SeatAvailability", width=60)
    self.ferry_table.column("From", width=60)
    self.ferry_table.column("To", width=60)
    self.ferry_table.column("Type", width=60)
    self.ferry_table.column("VehicleSpots", width=60)

    self.ferry_table.pack(fill=BOTH,expand=1)

    self.ferry_table.bind("<ButtonRelease-1>", self.get_cursor)

    self.display()



##FUNCTIONALITY DECLARATION

  def update(self):
    mydb = mysql.connector.connect(
      host="localhost",
      user="root",
      password="pes2ug20cs334",
      database="proj"
    )

    mycursor = mydb.cursor()
    mycursor.execute("UPDATE FERRY_INFO SET Ferry_id=%s,Ferry_name=%s,
Seat_availability=%s, source=%s , dest=%s, Type_of_ferry=%s, Vehicle_spots=%s ",
                     (self.FerryId.get(), self.Ferryname.get(),
                      self.Seat.get(), self.From.get(), self.To.get(),
```

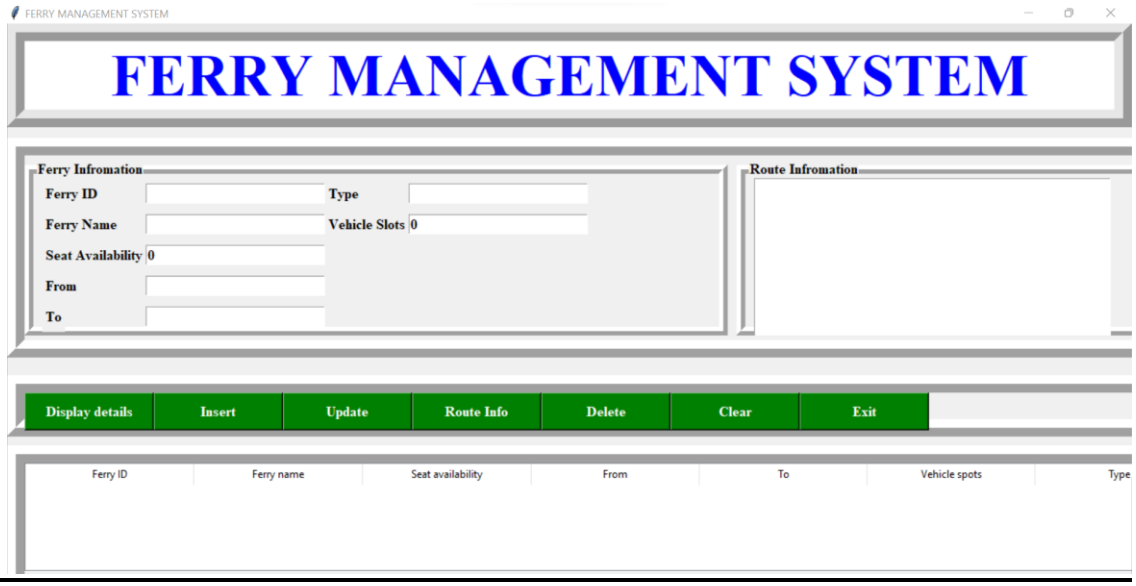
```
                 self.Type.get(), self.Slot.get())
                )
    mydb.commit()
    self.display()
    mydb.close()
    messagebox.showinfo("Success", "Record has been updated!")

  def insert(self):
    if self.FerryId.get()=="":
      messagebox.showerror("Error","FerryId is required!!")
    else:
      mydb = mysql.connector.connect(
        host="localhost",
        user="root",
        password="pes2ug20cs334",
        database="proj"
      )

      mycursor = mydb.cursor()

      mycursor.execute("INSERT INTO FERRY_INFO(Ferry_id,Ferry_name,
Seat_availability, source , dest , Type_of_ferry, Vehicle_spots)
VALUES(%s,%s,%s,%s,%s,%s,%s)",
                    (self.FerryId.get(),self.Ferryname.get(),
                     self.Seat.get(),self.From.get(),self.To.get(),
                     self.Type.get(),self.Slot.get()))
      mydb.commit()
      self.display()
      mydb.close()
      messagebox.showinfo("Success","Record has been inserted!")



  def information(self):
    mydb = mysql.connector.connect(
      host="localhost",
      user="root",
      password="pes2ug20cs334",
      database="proj"
    )

    mycursor = mydb.cursor()
    self.txtInformation.insert(END,"Ferry ID:\t\t\t" + self.FerryId.get() + "\n")
```

```
  self.txtInformation.insert(END, "Ferry Name:\t\t\t" + self.Ferryname.get() + "\n")
  self.txtInformation.insert(END, "Source:\t\t\t" + self.From.get() + "\n")
  self.txtInformation.insert(END, "Destination:\t\t\t" + self.To.get() + "\n")
  query = "SELECT Travel_time FROM ROUTE_INFO WHERE Ferry_id = %s"
  value=(self.FerryId.get(),)
  mycursor.execute(query,value)
  res=mycursor.fetchall()
  self.txtInformation.insert(END, "Travel time:\t\t\t" ,res , "\n")
  mydb.commit()
  mydb.close()


def display(self):
    mydb = mysql.connector.connect(
      host="localhost",
      user="root",
      password="pes2ug20cs334",
      database="proj"
    )

    mycursor = mydb.cursor()
    query = "SELECT * FROM FERRY_INFO WHERE Ferry_id = %s"
    value = (self.FerryId.get(),)
    mycursor.execute(query, value)
    res = mycursor.fetchall()
    if len(res) != 0:
      self.ferry_table.delete(*self.ferry_table.get_children())
      for i in res:
        self.ferry_table.insert("", END, values=i)
      mydb.commit()
    mydb.close()

def delete(self):
  mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="pes2ug20cs334",
    database="proj"
  )

  mycursor = mydb.cursor()
  query = "DELETE  FROM FERRY_INFO WHERE Ferry_id = %s"
  value = (self.FerryId.get(),)
```

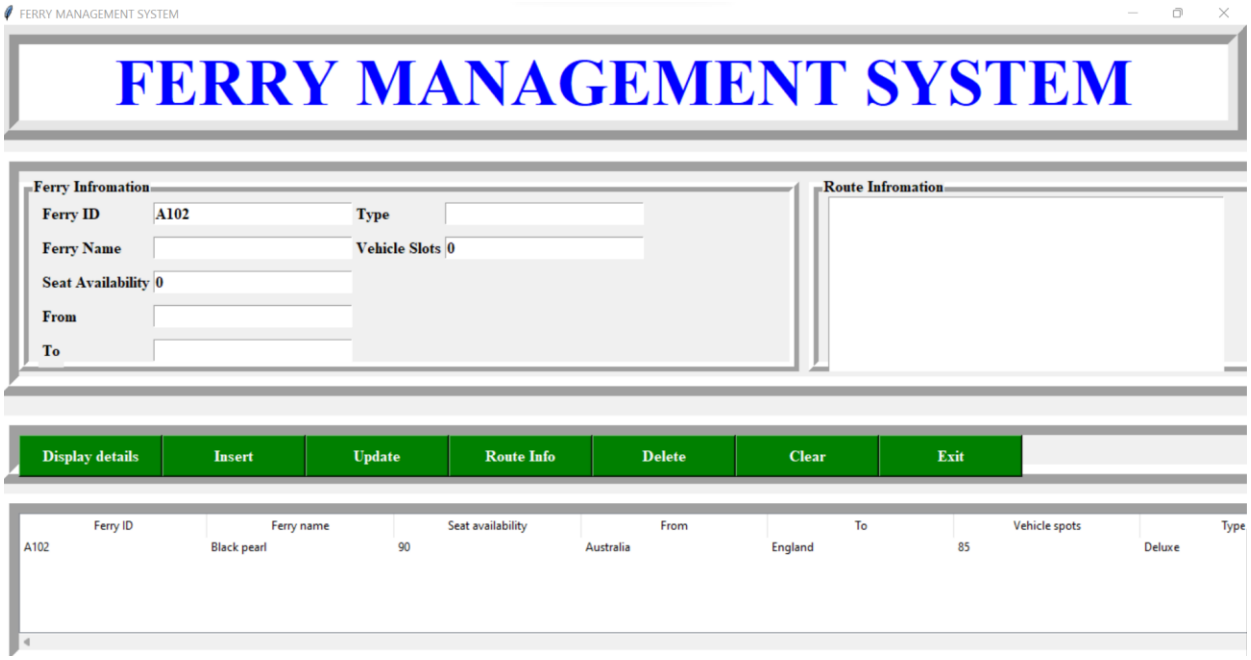
```
    mycursor.execute(query,value)
    mydb.commit()
    mydb.close()
    messagebox.showinfo("Delete", "Record has been Deleted!")



  def exit(self):
    exit=messagebox.askyesno("Ferry Management System","Do you want to exit?")
    if exit>0:
      root.destroy()
      return


  def clear(self):
    self.FerryId.set("")
    self.Ferryname.set("")
    self.Seat.set("")
    self.From.set("")
    self.To.set("")
    self.Type.set("")
    self.Slot.set("")
    self.txtInformation.delete("1.0",END)


  def get_cursor(self,event=""):
    cursor_row=self.ferry_table.focus()
    content=self.ferry_table.item(cursor_row)
    row=content["values"]
    self.FerryId.set(row[0])
    self.Ferryname.set(row[1])
    self.Seat.set(row[2])
    self.From.set(row[3])
    self.To.set(row[4])
    self.Type.set(row[5])
    self.Slot.set(row[6])



root=Tk()
ob=ferry(root)
root.mainloop()
```

OUTPUT SCREENSHOTS:

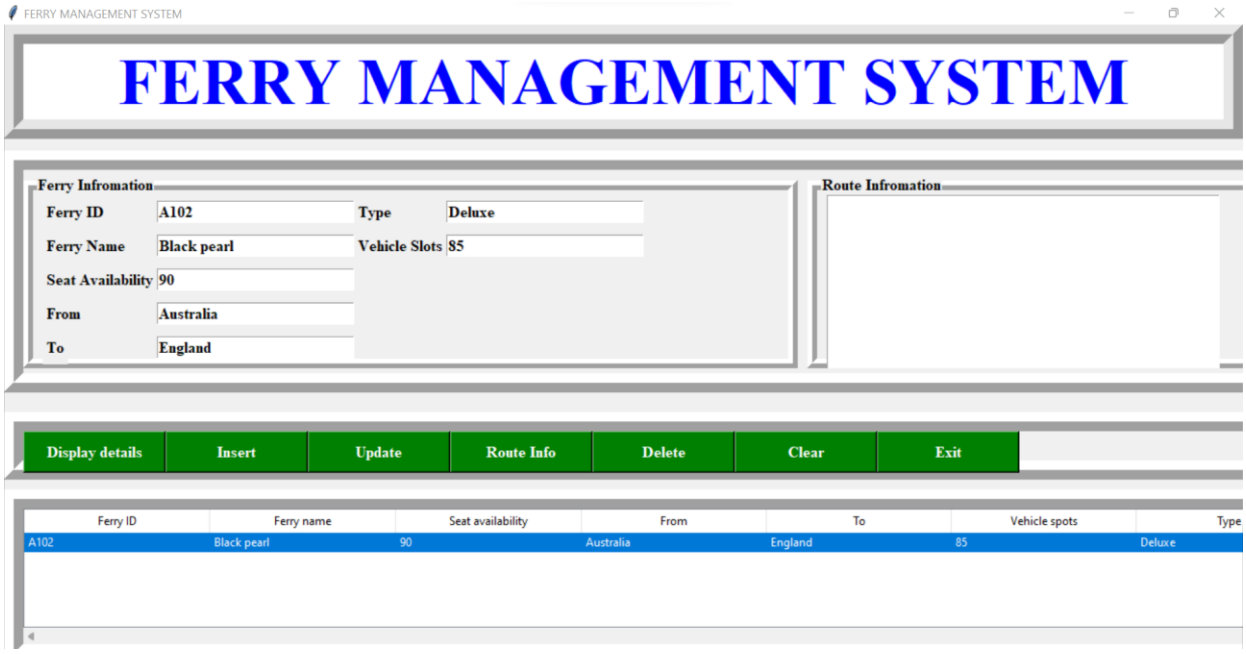
1) Initial Window



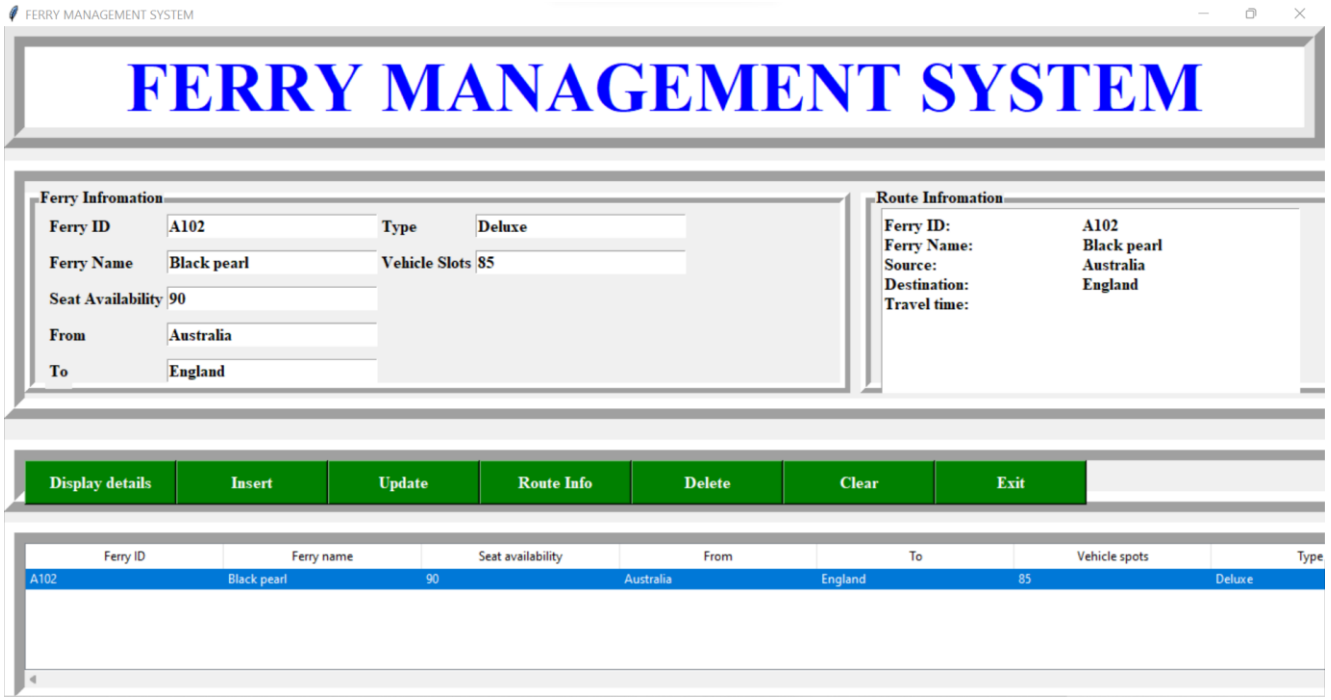
2) Displaying ferry details



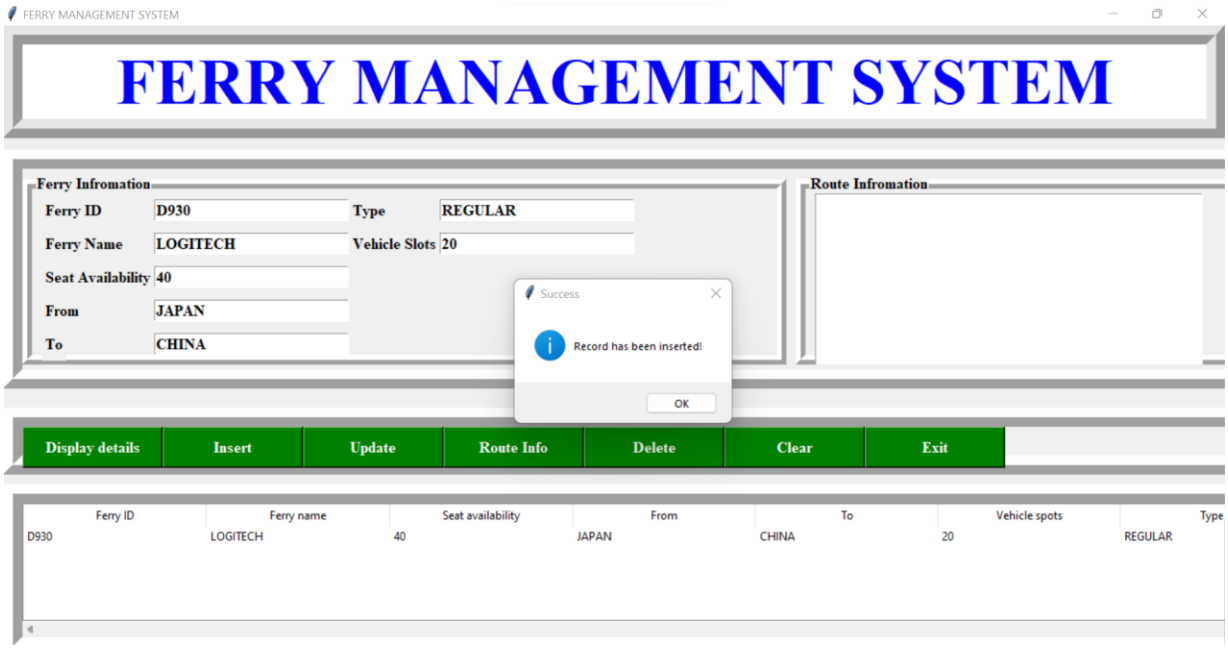
3)


FERRY MANAGEMENT SYSTEM
FERRY MANAGEMENT SYSTEM
Ferry Infromation
Ferry ID
A102
Type
Deluxe
Ferry Name
Black pearl
Vehicle Slots
85
Seat Availability
90
From
Australia
To
England
Route Infromation
Display details
Insert
Update
Route Info
Delete
Clear
Exit
Ferry ID
Ferry name
Seat availability
From
To
Vehicle spots
Type
A102
Black pearl
90
Australia
England
85
Deluxe

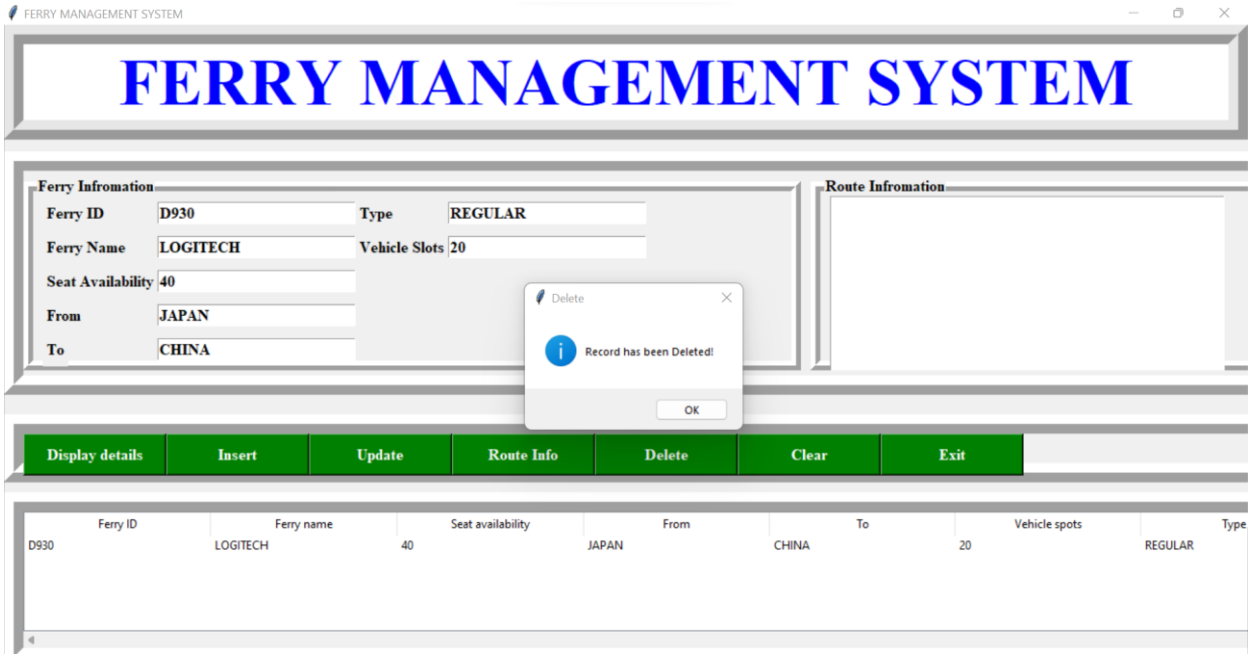
4)


FERRY MANAGEMENT SYSTEM
FERRY MANAGEMENT SYSTEM
Ferry Infromation
Ferry ID
A102
Type
Deluxe
Ferry Name
Black pearl
Vehicle Slots
85
Seat Availability
90
From
Australia
To
England
Route Infromation
Ferry ID:
A102
Ferry Name:
Black pearl
Source:
Australia
Destination:
England
Travel time:
Display details
Insert
Update
Route Info
Delete
Clear
Exit
Ferry ID
Ferry name
Seat availability
From
To
Vehicle spots
Type
A102
Black pearl
90
Australia
England
85
Deluxe

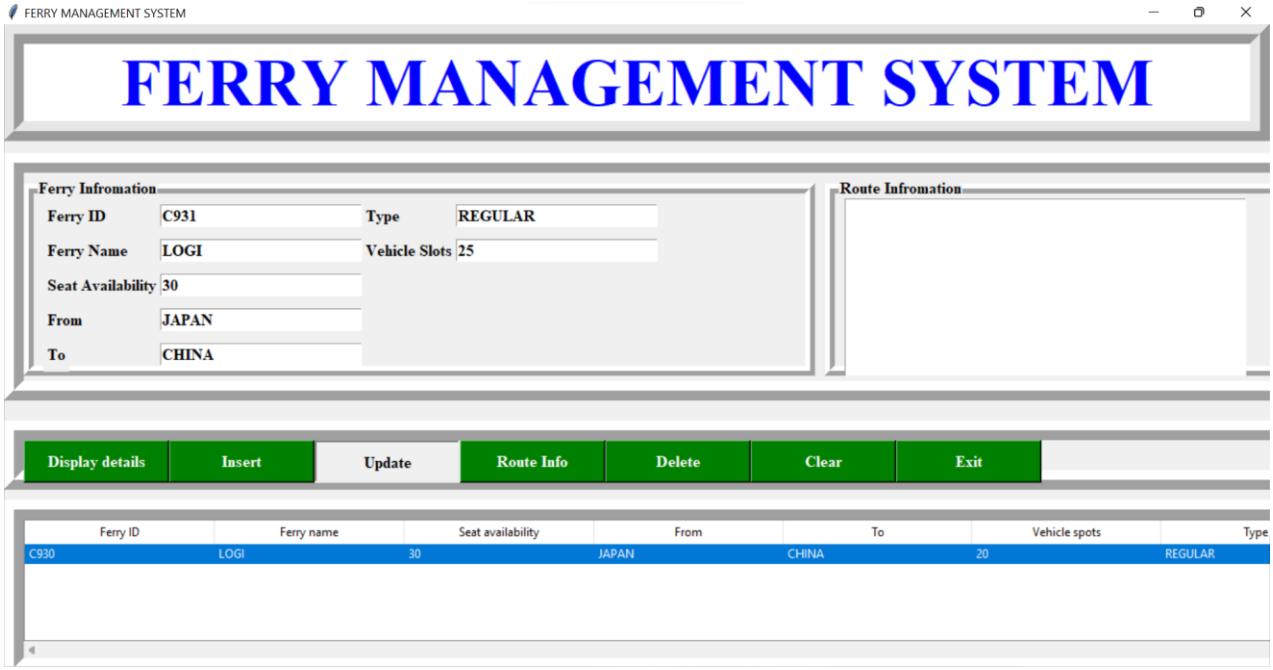
5) Inserting record



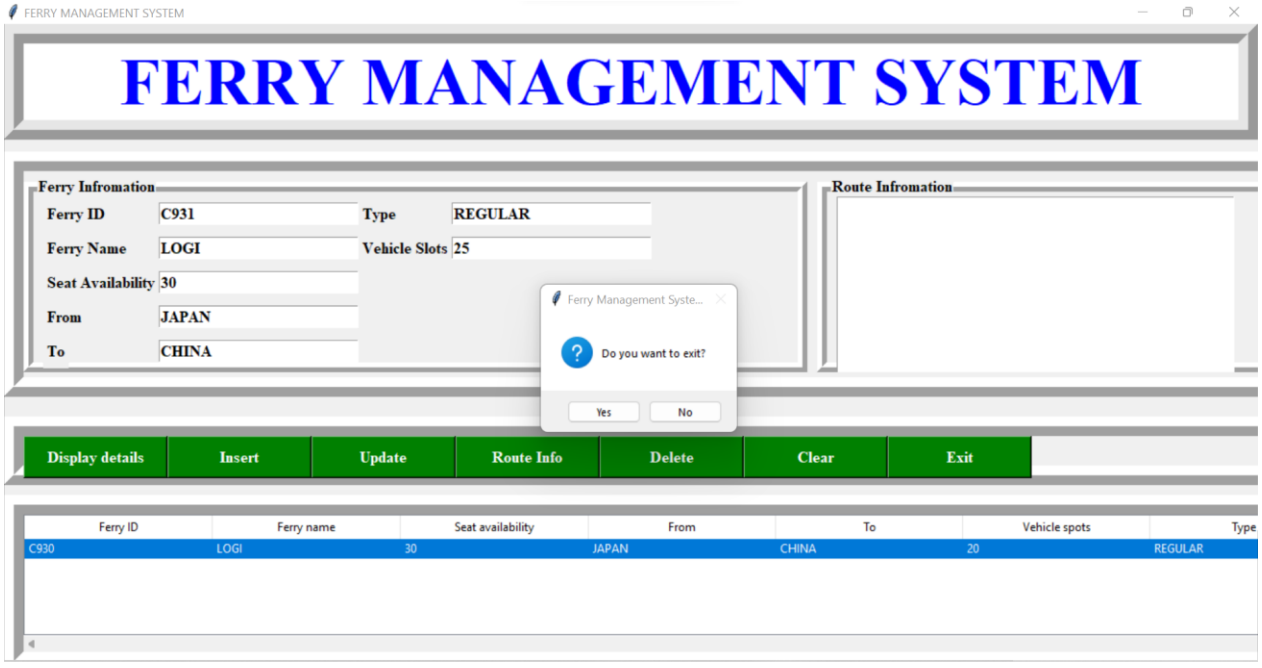
6) Deleting



7) Updating



8) Exit



12. Conclusion

The ferry management system will be helpful for the admin to keep track of the ferries and corresponding information associated with each ferry.

For the future scope of the project, it can be used to book and cancel tickets as well with respect to the frontend.

The project has been created using a Python and MYSQL interface.