AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)

FACULTY OF SCIENCE & TECHNOLOGY

Advance Database Management System

Metro Rail Management System

Section: A

Supervised By

Juena Ahmed Noshin

Submitted By

1. Md. Amir Hossain Alif
2. Asir Foysal Al Mukit
3. Umme Jannatul Fariha
4. Afsana Jahan Onu

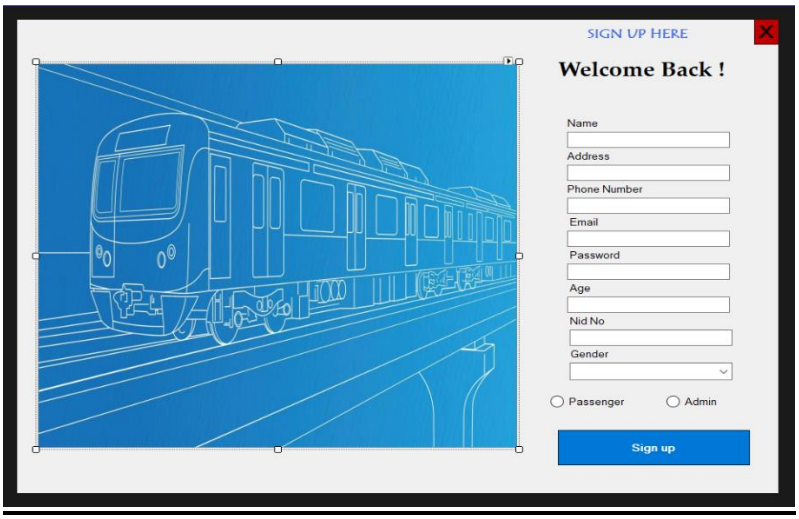
TABLE OF CONTENTS

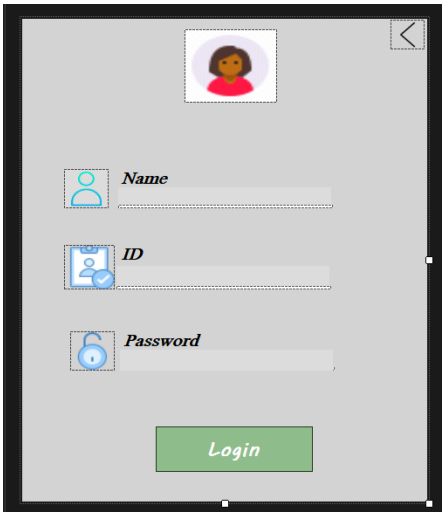
Introduction

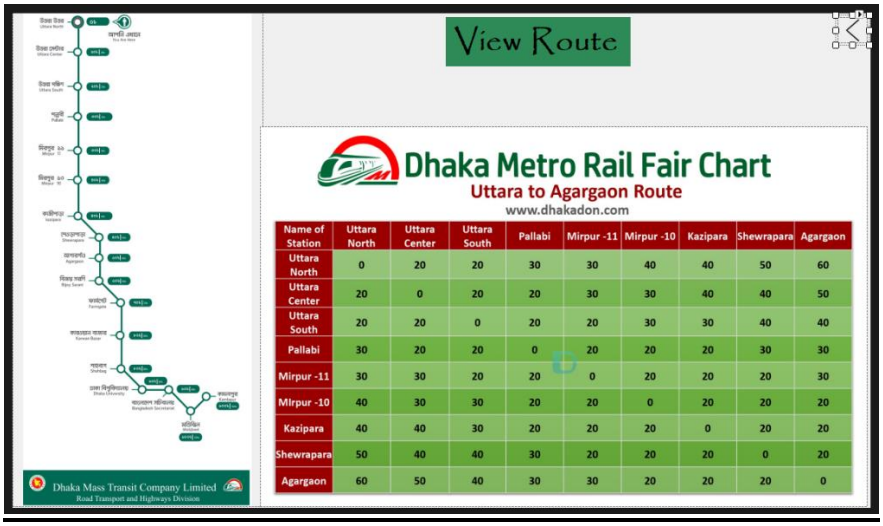
The project deals with designing and implementing a Metro Station Management System for managing all activities involving the metro. This would include ticketing, passenger information, train schedules, route management, and administration. The proposed system will be designed with improved operation efficiency and convenient and smooth travel for the passengers with various facilities such as rapid buying of tickets, up-to-date information, and route management on interactive pages.

The system will integrate core entities, such as passengers, tickets, stations, routes, transactions, and staff, for accurate data management and smooth coordination toward optimum service delivery. It caters to both passengers and administrators by allowing appropriate tracking of data in real time and optimizing the overall service.

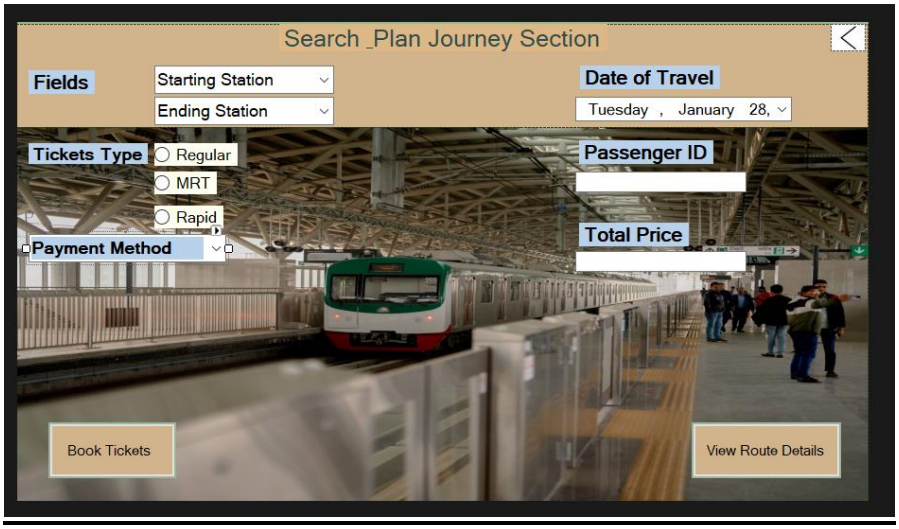
User Interface



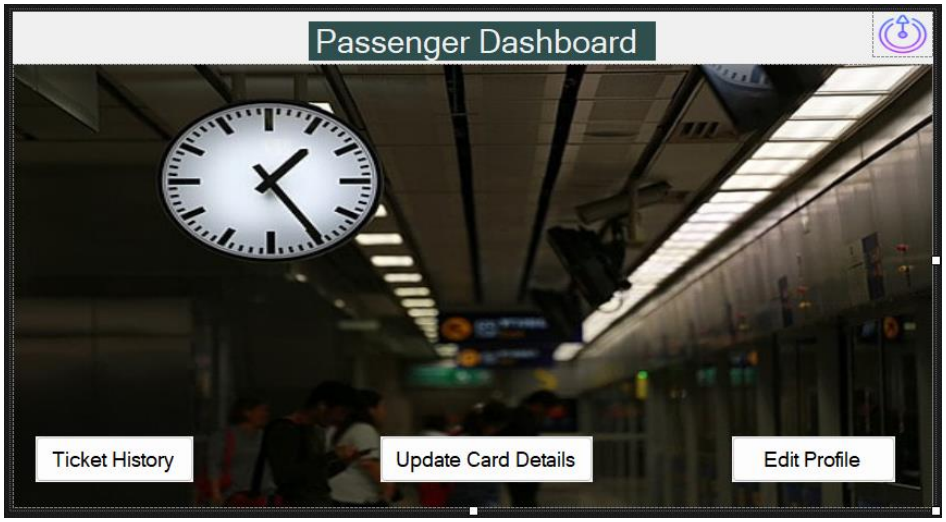


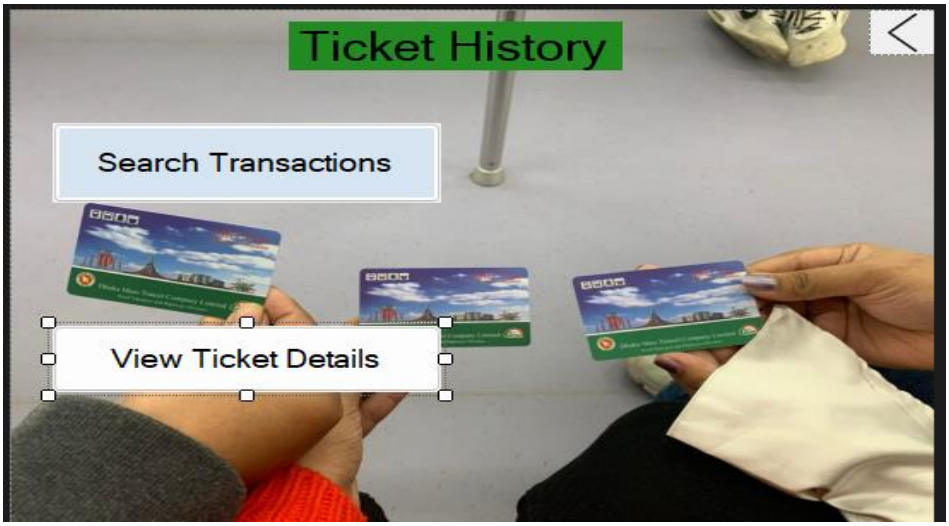


Name of Station	Uttara North	Uttara Center	Uttara South	Pallabi	Mirpur -11	Mirpur -10	Kazipara	Shewrapara	Agargaon
Uttara North	0	20	20	30	30	40	40	50	60
Uttara Center	20	0	20	20	30	30	40	40	50
Uttara South	20	20	0	20	20	30	30	40	40
Pallabi	30	20	20	0	20	20	20	30	30
Mirpur -11	30	30	20	20	0	20	20	20	30
MIrpur -10	40	30	30	20	20	0	20	20	20
Kazipara	40	40	30	20	20	20	0	20	20
Shewrapara	50	40	40	30	20	20	20	0	20
Agargaon	60	50	40	30	30	20	20	20	0

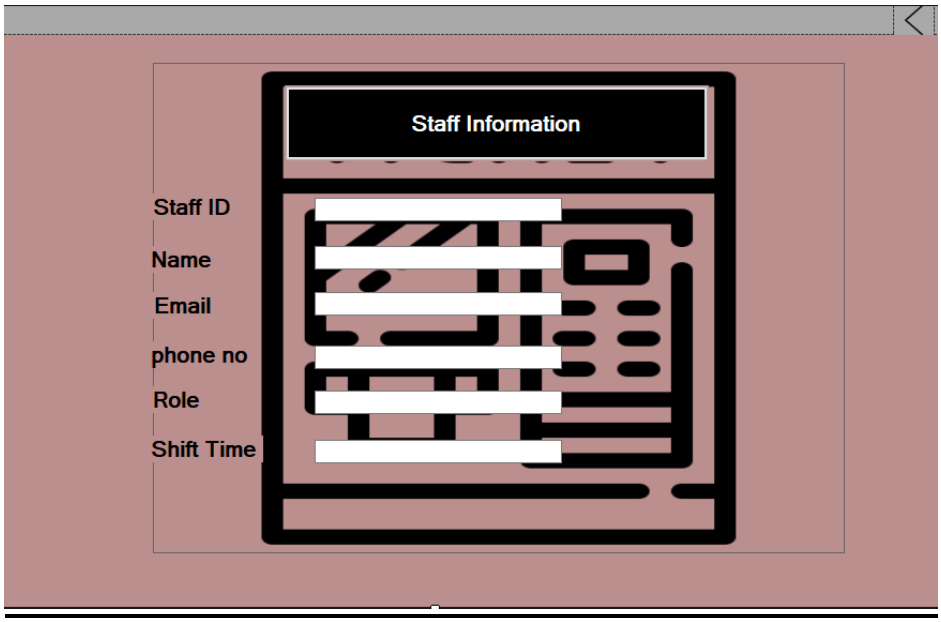



Card Details
Card Type
Card ID
Balance
Issue Date
Exp Date


Passenger Dashboard
Ticket History
Update Card Details
Edit Profile


Ticket History
Search Transactions
View Ticket Details


Admin Panel
Passenger Management
Staff Management
Route Management
Train Management


Staff Information
Staff ID
Name
Email
phone no
Role
Shift Time

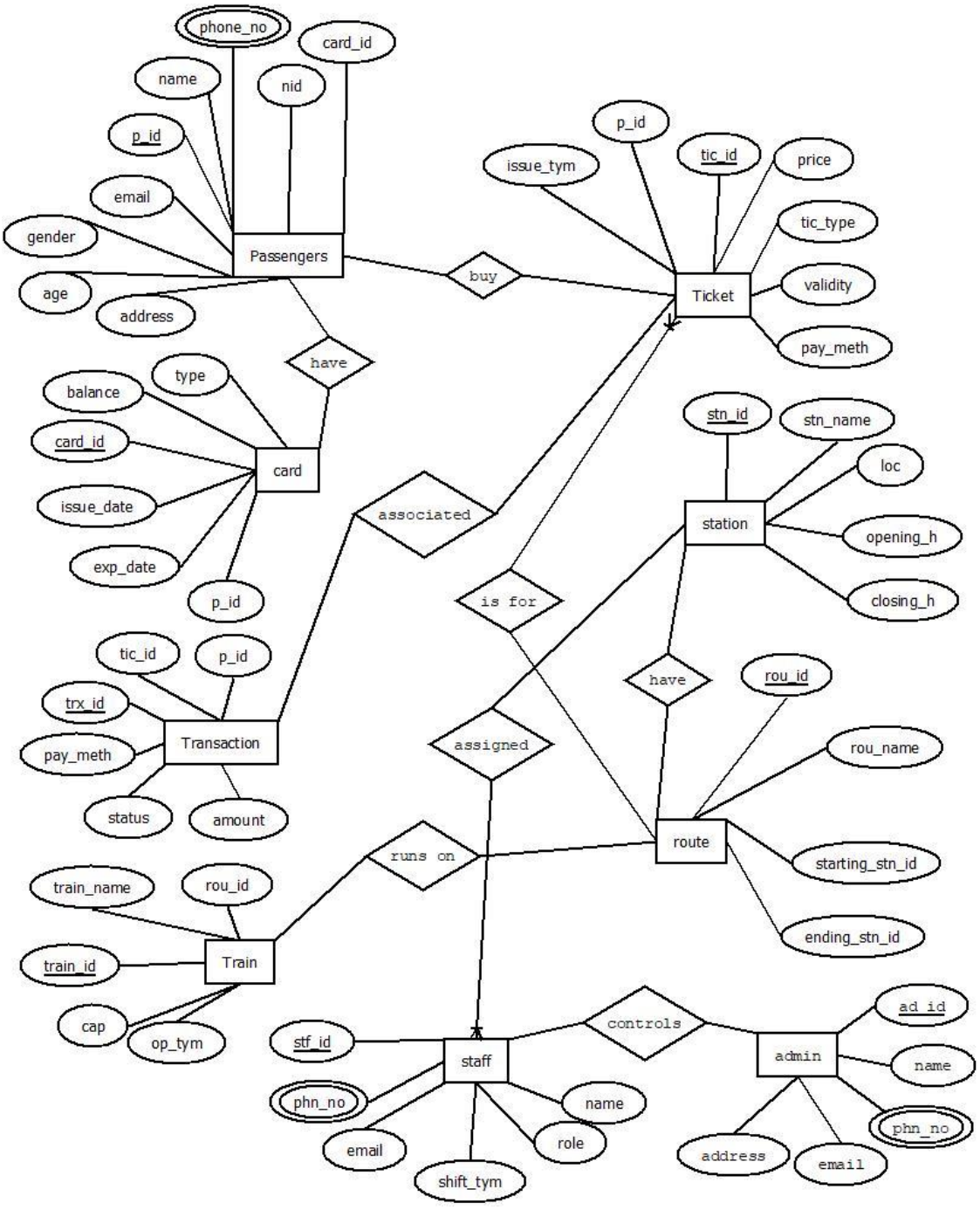
Scenario Description

The metro rail management system efficiently manages passenger ticketing, train operations, and station staffing. Passengers can register in the system with a unique passenger ID and provide details such as name, contact information, address, age, gender, and national identification. Each passenger is issued a single metro card containing a card ID, type, balance, issue date, and expiration date, which can be used for purchasing tickets. Tickets, identified by a unique ticket ID, include details such as type, issue time, validity, and payment method. Passengers can buy multiple tickets, with each ticket linked to a specific transaction. Transactions, identified by transaction ID, record the passenger ID, ticket ID, payment method, status, and amount. Tickets are associated with predefined routes, comprising a route ID, name, starting station, and ending station. Trains operate on these routes and are defined by train ID, name, route ID, capacity, and operating hours. Each route encompasses multiple stations, which are identified by station ID, name, location, and operational hours. The system is managed by admins who assign staff to stations, ensuring efficient operations. Admins are identified by admin ID and maintain their contact information and address. Stations are staffed by personnel with specific roles and shift times, with each staff member assigned to one station. This integrated system ensures seamless ticketing, route management, and operational efficiency for passengers and administrators alike.

Project Proposal

Metro Rail Management System is a digital solution for the betterment of railway operations by enabling passengers to purchase tickets online, maintain their travel cards, and manage transactions efficiently. The system features include passenger registration, secure ticket purchase, smart card integration, and automated financial transaction processing. It also includes train and route management, and administrative control for staff and operators. The system is focused on digitalizing railway ticketing, reducing manual workload, and enhancing passenger experience by using C# for both frontend and backend, and Oracle for the database. This project ensures a fast, seamless, and secure railway management system, hence guaranteeing efficiency, security, and user satisfaction with secure payment gateways and automated processes.

ER Diagram


phone_no
card_id
name
nid
p_id
email
gender
Passengers
age
address
buy
p_id
issue_tym
tic_id
price
tic_type
validity
pay_meth
Ticket
have
type
balance
card_id
card
issue_date
exp_date
p_id
associated
stn_id
stn_name
loc
station
opening_h
closing_h
is for
tic_id
p_id
trx_id
pay_meth
Transaction
status
amount
assigned
have
rou_id
rou_name
route
starting_stn_id
ending_stn_id
runs on
train_name
rou_id
train_id
Train
cap
op_tym
stf_id
staff
phn_no
name
email
role
shift_tym
controls
ad_id
admin
name
phn_no
address
email

Normalization

BUY:

UNF:

buy(p_id,name,nid,phn_no,email,age,gender,address,card_id,tic_id,tic_type,issue_tym,price,validity, pay_meth,p_id)

1NF:

Phn_no is a multivalued attribute.

1. p_id,name,nid,phn_no,email,age,gender,address,card_id,tic_id,tic_type,issue_tym,price,validity, pay_meth,p_id

2NF:

1. p_id,name,nid,phn_no,email,age,gender,address,card_id
2. tic_id,tic_type,issue_tym,price,validity,pay_meth,p_id

3NF:

There is no transitive dependency. Relation already in 3NF.

1. p_id,name,nid,phn_no,email,age,gender,address,card_id
2. tic_id,tic_type,issue_tym,price,validity,pay_meth,p_id

Table Creation:

1. p_id,name,nid,phn_no,email,age,gender,address,card_id
2. tic_id,tic_type,issue_tym,price,validity,pay_meth,p_id

ASSOCIATED:

UNF:

associated(trx_id,tic_id,p_id,pay_meth,status,amount,tic_id,tic_type,issue_tym,price,validity,pay_meth, p_id)

1NF:

There is no multivalued attribute. Relation already in 1nf.

1.trx_id,tic_id,p_id,pay_meth,status,amount,tic_id,tic_type,issue_tym,price,validity,pay_meth,p_id

2NF:

1.trx_id,tic_id,p_id,pay_meth,status,amount

2.tic_id,tic_type,issue_tym,price,validity,pay_meth,p_id

3NF:

There is no transitive dependency. Relation already in 3NF.

1.trx_id,tic_id,p_id,pay_meth,status,amount

2.tic_id,tic_type,issue_tym,price,validity,pay_meth,p_id

Table Creation:

1.trx_id,tic_id,p_id,pay_meth,status,amount

2.tic_id,tic_type,issue_tym,price,validity,pay_meth,p_id

HAVE:

UNF:

have(sttn_id,sttn_name,loc,opening_h,closing_h,rou_id,rou_name,strt_sttn_id,end_sttn_id)

1NF:

There is no multivalued attribute. Relation already in 1nf.

1. sttn_id,sttn_name,loc,opening_h,closing_h,rou_id,rou_name,strt_sttn_id,end_sttn_id

2NF:

1.sttn_id,sttn_name,loc,opening_h,closing_h

2.rou_id,rou_name,strt_sttn_id,end_sttn_id

3NF:

There is no transitive dependency. Relation already in 3NF.

1.sttn_id,sttn_name,loc,opening_h,closing_h

2.rou_id,rou_name,strt_sttn_id,end_sttn_id

Table Creation:

1.sttn_id,sttn_name,loc,opening_h,closing_h

2.rou_id,rou_name,strt_sttn_id,end_sttn_id

RUNS_ON:

UNF:

runs_on(train_id,train_name,rou_id,cap,op_tym,rou_id,rou_name,strt_sttn_id,end_sttn_id)

1NF:

There is no multivalued attribute. Relation already in 1nf.

1.train_id,train_name,rou_id,cap,op_tym,rou_id,rou_name,strt_sttn_id,end_sttn_id

2NF:

1.train_id,train_name,rou_id,cap,op_tym

2.rou_id,rou_name,strt_sttn_id,end_sttn_id

3NF:

There is no transitive dependency. Relation already in 3NF.

1.train_id,train_name,rou_id,cap,op_tym

2.rou_id,rou_name,strt_sttn_id,end_sttn_id

Table Creation:

1.train_id,train_name,rou_id,cap,op_tym

2.rou_id,rou_name,strt_sttn_id,end_sttn_id

ASSIGNED:

UNF:

assigned(sttn_id,sttn_name,loc,opening_h,closing_h,staff_id,name,role,phn_no,email,shift_tym,assigned_sttn_id)

1NF:

Phn_no is a multivalued attribute.

1.sttn_id,sttn_name,loc,opening_h,closing_h,staff_id,name,role,phn_no,email,shift_tym,assigned_sttn_id

2NF:

1.sttn_id,sttn_name,loc,opening_h,closing_h

2. staff_id,name,role,phn_no,email,shift_tym,assigned_sttn_id

3NF:

There is no transitive dependency. Relation already in 3NF.

1.sttn_id,sttn_name,loc,opening_h,closing_h

2. staff_id,name,role,phn_no,email,shift_tym,assigned_sttn_id

Table Creation:

1.sttn_id,sttn_name,loc,opening_h,closing_h

2. staff_id,name,role,phn_no,email,shift_tym,assigned_sttn_id

HAVE_CARD:

UNF:

Have_card(p_id,name,nid,phn_no,email,age,gender,address,card_id,card_id,type,balance,issue_date, exp_date,p_id)

1NF:

Phn_no is a multivalued attribute.

1.p_id,name,nid,phn_no,email,age,gender,address,card_id,card_id,type,balance,issue_date, exp_date,p_id

2NF:

1. p_id,name,nid,phn_no,email,age,gender,address,card_id

2. card_id,type,balance,issue_date,exp_date,p_id

3NF:

There is no transitive dependency. Relation already in 3NF.

1. p_id,name,nid,phn_no,email,age,gender,address,card_id

2. card_id,type,balance,issue_date,exp_date,p_id

Table Creation:

1. p_id,name,nid,phn_no,email,age,gender,address,card_id

2. card_id,type,balance,issue_date,exp_date,p_id

CONTROL:

UNF:

control(ad_id,

name,phn_no,email,addrs,nid,staff_id,name,role,phn_no,email,shift_tym,assigned_sttn_id)

1NF:

Phn_no is a multivalued attribute.

1.ad_id, name,phn_no,email,addrs,nid,staff_id,name,role,phn_no,email,shift_tym,assigned_sttn_id

2NF:

1.ad_id, name,phn_no,email,addrs,nid

2. staff_id,name,role,phn_no,email,shift_tym,assigned_sttn_id

3NF:

There is no transitive dependency. Relation already in 3NF.

1.ad_id, name,phn_no,email,addrs,nid

2. staff_id,name,role,phn_no,email,shift_tym,assigned_sttn_id

Table Creation:

1.ad_id, name,phn_no,email,addrs,nid

2. staff_id,name,role,phn_no,email,shift_tym,assigned_sttn_id

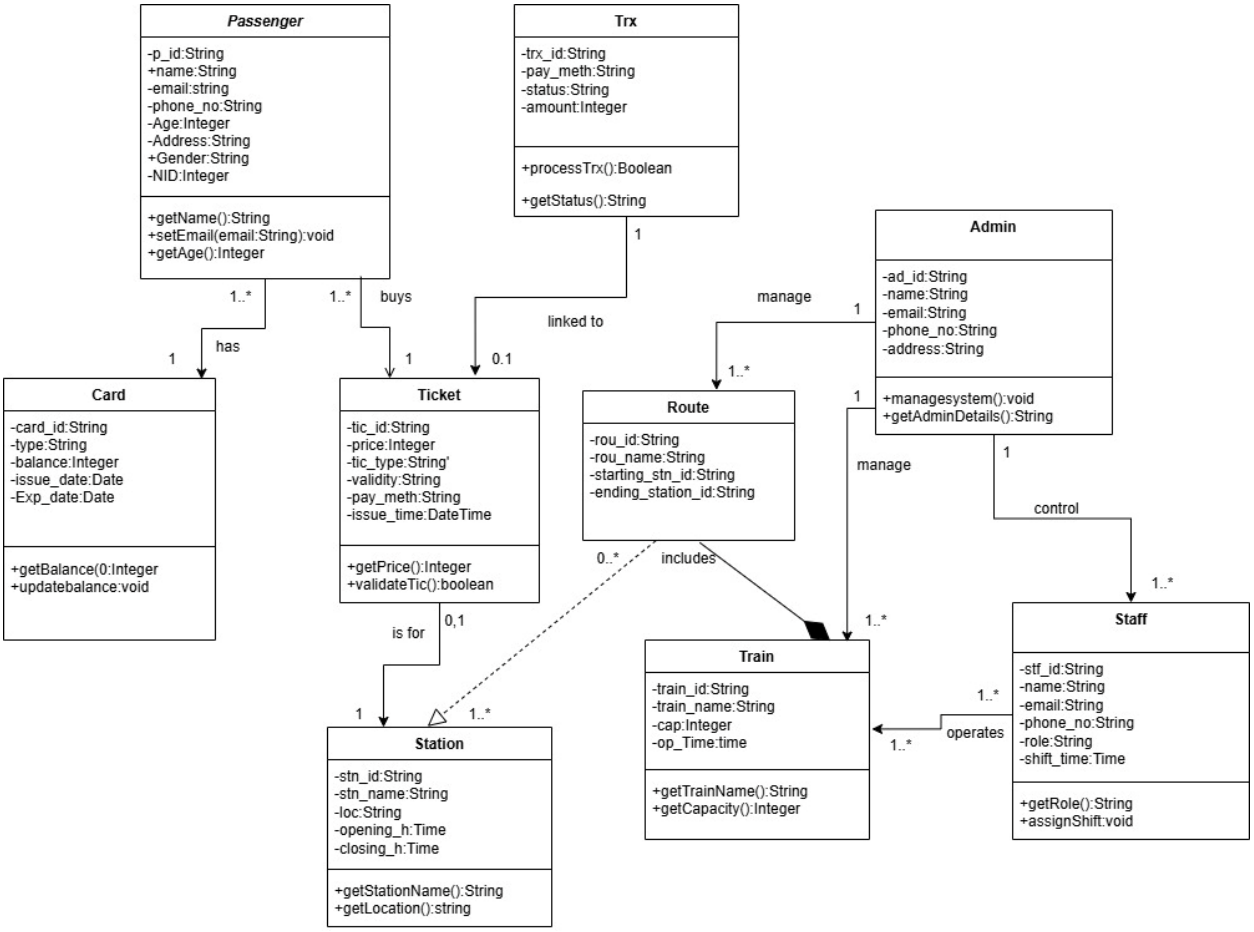
Temporary Tables:

01. p_id,name,nid,phn_no,email,age,gender,address,card_id
02. tic_id,tic_type,issue_tym,price,validity,pay_meth,p_id
03.trx_id,tic_id,p_id,pay_meth,status,amount
~~04.tic_id,tic_type,issue_tym,price,validity,pay_meth,p_id~~
05.sttn_id,sttn_name,loc,opening_h,closing_h
06.rou_id,rou_name,strt_sttn_id,end_sttn_id
07.train_id,train_name,rou_id,cap,op_tym
~~08.rou_id,rou_name,strt_sttn_id,end_sttn_id~~
~~09.sttn_id,sttn_name,loc,opening_h,closing_h~~
10. staff_id,name,role,phn_no,email,shift_tym,assigned_sttn_id
~~11. p_id,name,nid,phn_no,email,age,gender,address,card_id~~
12. card_id,type,balance,issue_date,exp_date,p_id
13.ad_id, name,phn_no,email,addrs,nid
~~14. staff_id,name,role,phn_no,email,shift_tym,assigned_sttn_id~~

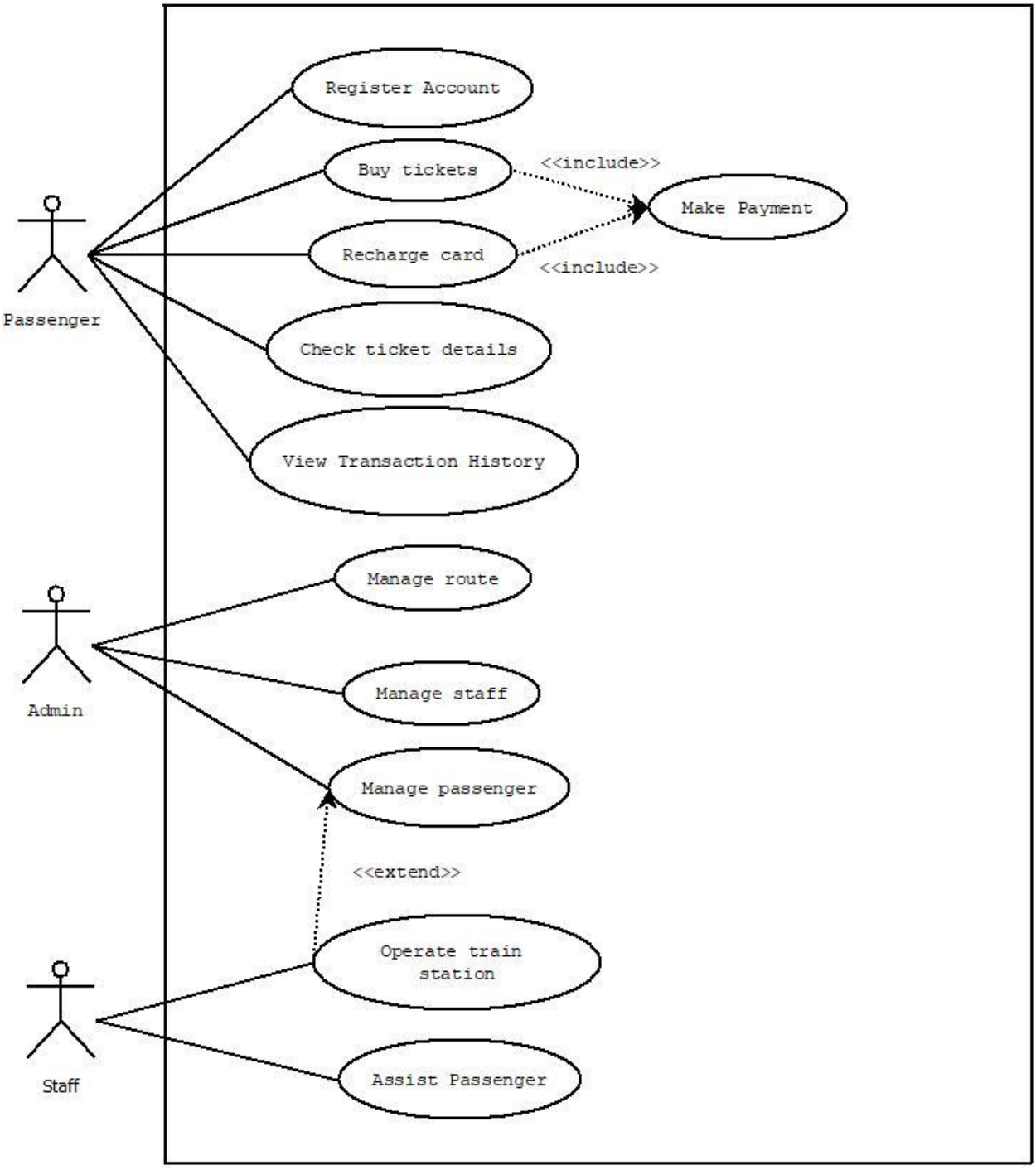
Final Tables:

01. p_id,name,nid,phn_no,email,age,gender,address,card_id
02. tic_id,tic_type,issue_tym,price,validity,pay_meth,p_id
03.trx_id,tic_id,p_id,pay_meth,status,amount
04.sttn_id,sttn_name,loc,opening_h,closing_h
05.rou_id,rou_name,strt_sttn_id,end_sttn_id
06.train_id,train_name,rou_id,cap,op_tym
07. staff_id,name,role,phn_no,email,shift_tym,assigned_sttn_id
08. card_id,type,balance,issue_date,exp_date,p_id
09.ad_id, name,phn_no,email,addrs,nid

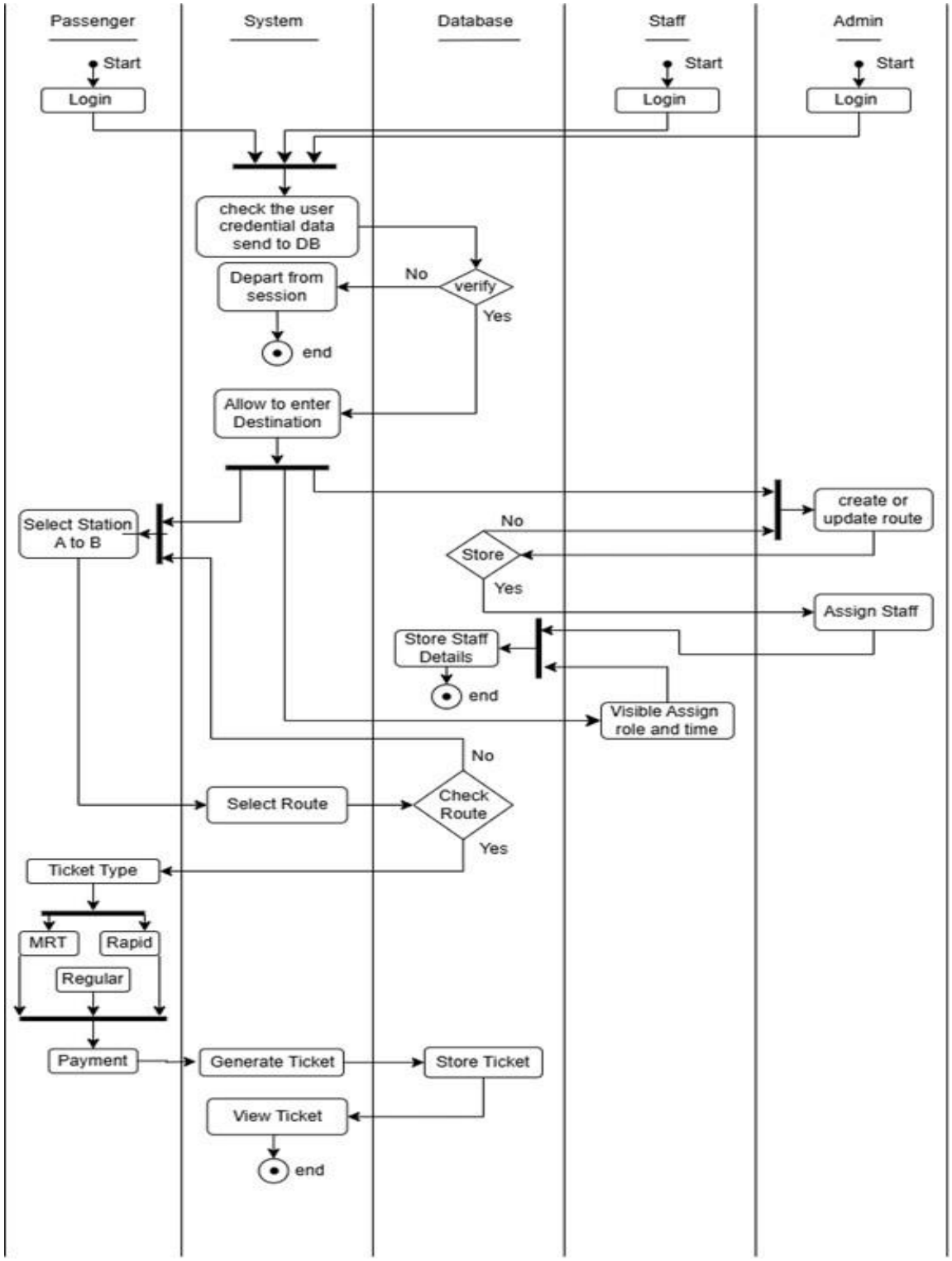
Class diagram:



Use Case Diagram:



Activity Diagram:



Schema Diagram:

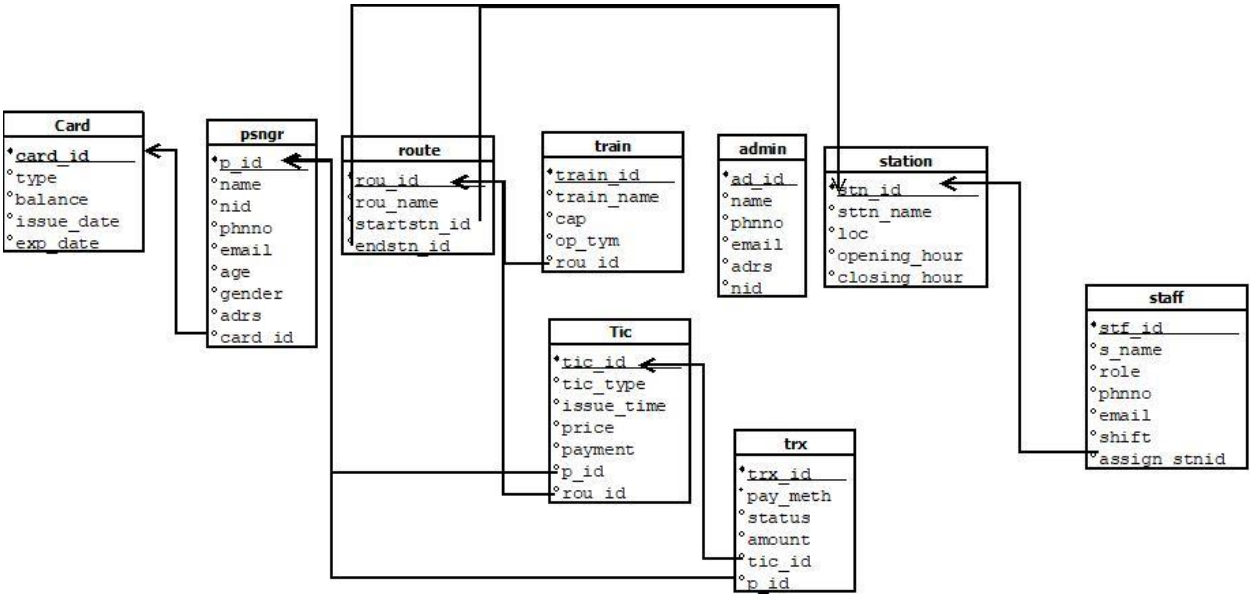
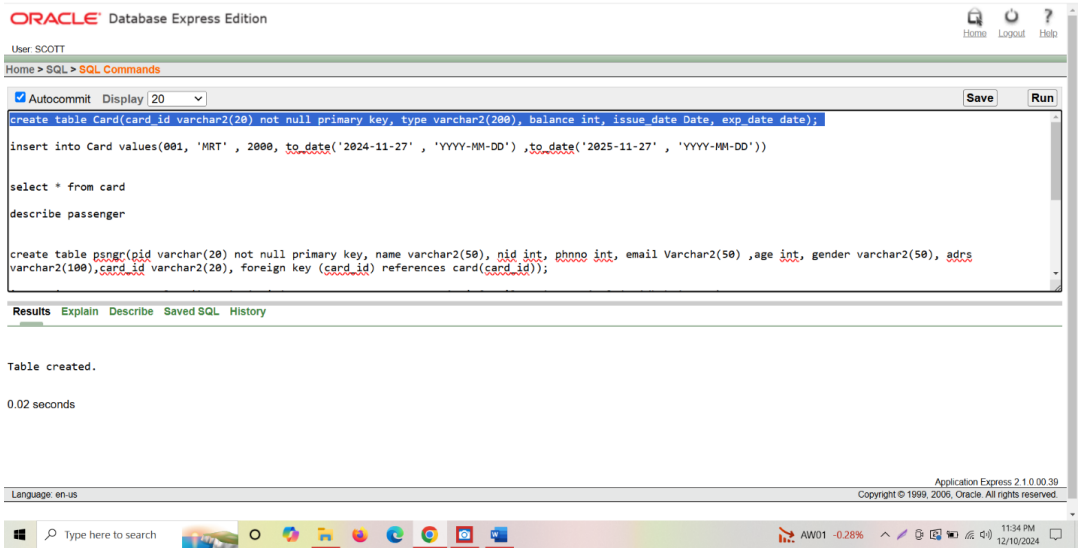
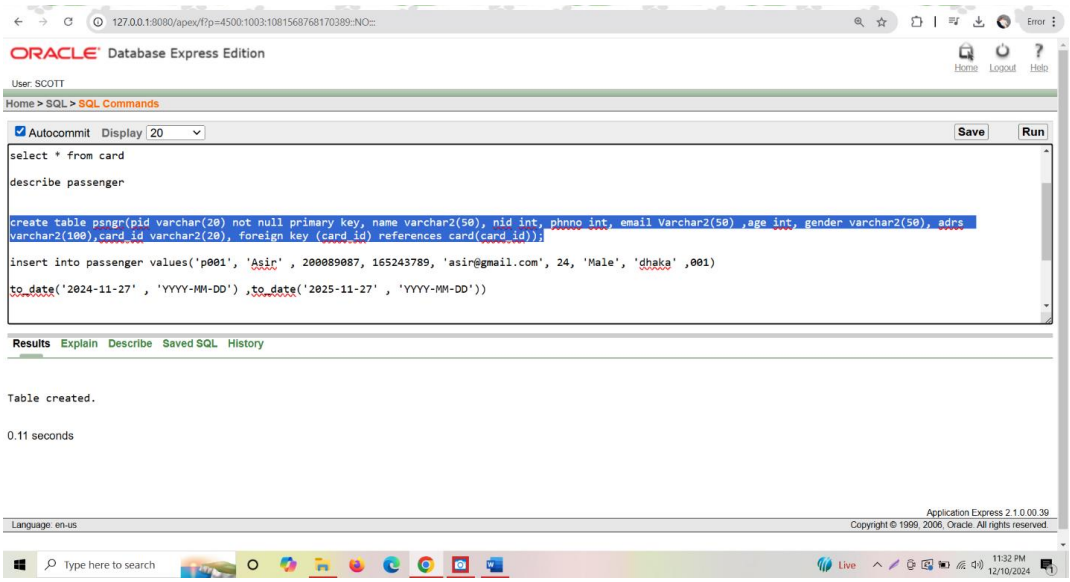


TABLE CREATION

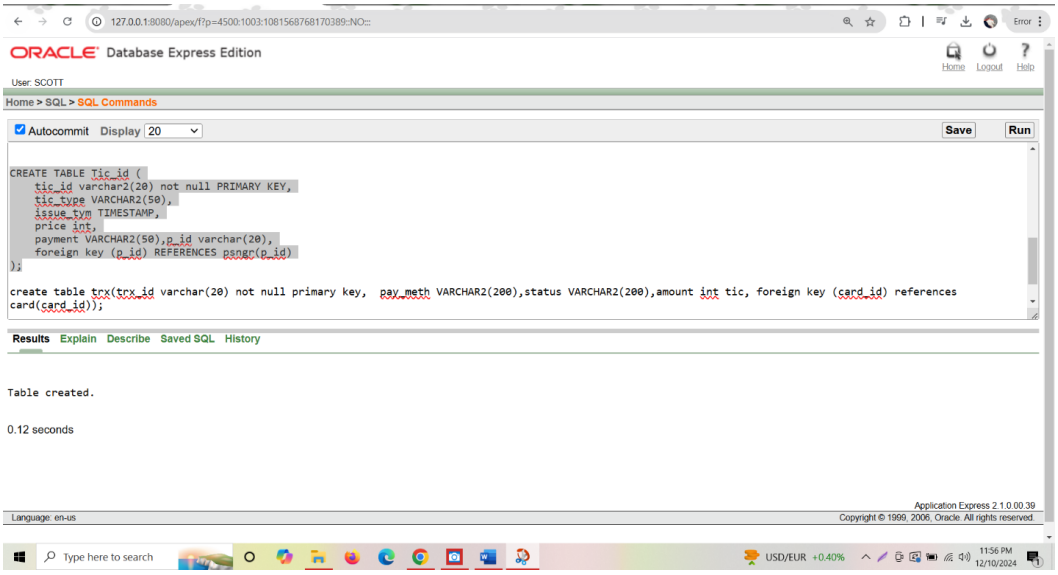
1. Card:
   create table Card(card_id varchar2(20) not null primary key, type varchar2(200), balance int, issue_date Date, exp_date date);



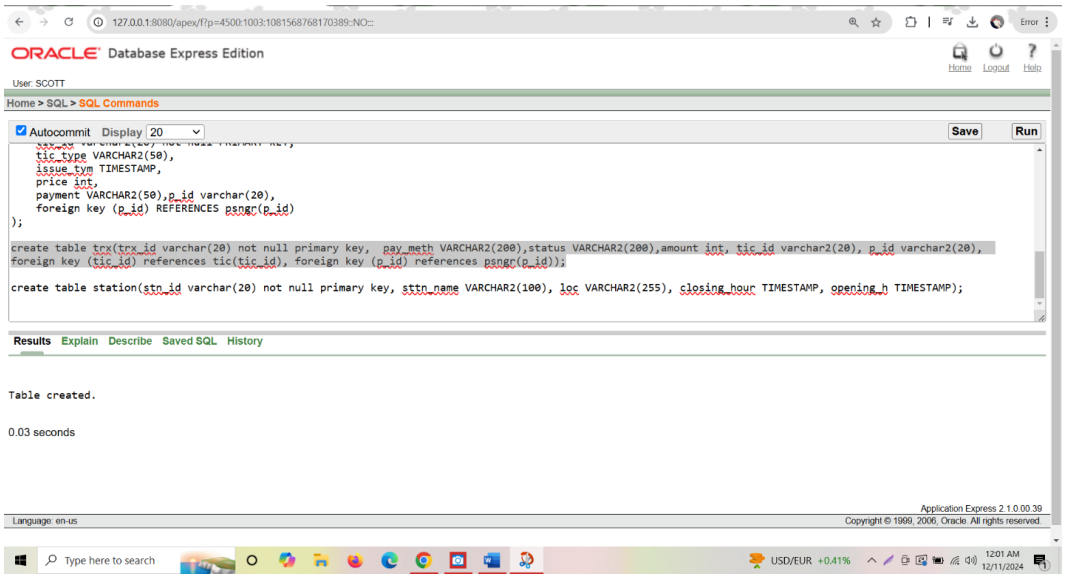
2. Psngr:
   create table psngr(p_id varchar(20) not null primary key, name varchar2(50), nid int, phnno int, email Varchar2(50) ,age int, gender varchar2(50), adrs varchar2(100),card_id varchar2(20), foreign key (card_id) references card(card_id));



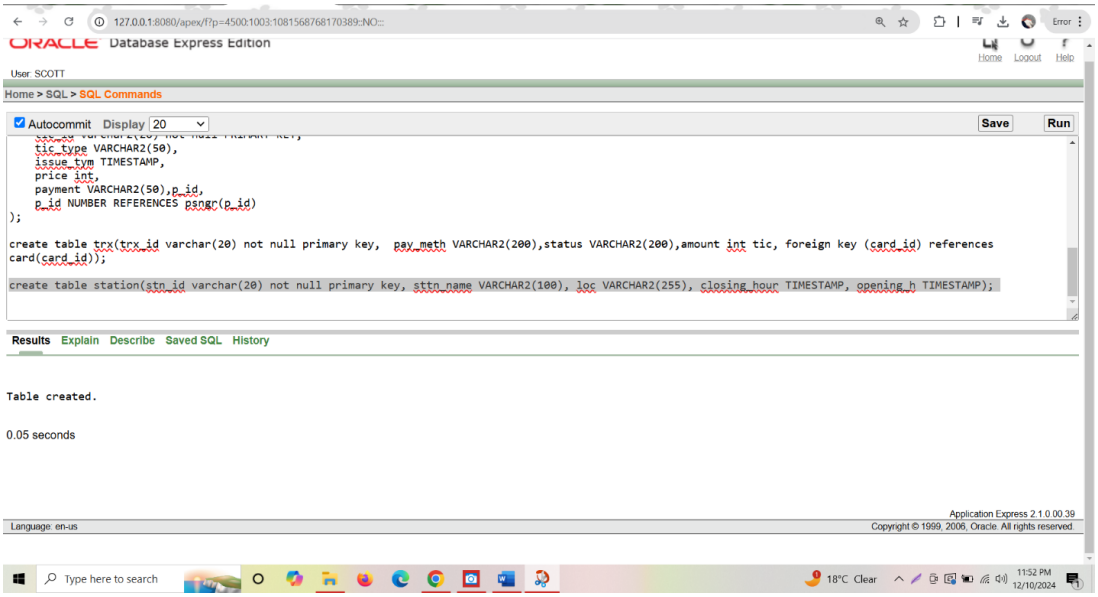
3. Ticket:
   CREATE TABLE Tic (tic_id varchar2(20) not null PRIMARY KEY, tic_type VARCHAR2(50), issue_tym TIMESTAMP, price int, payment VARCHAR2(50),p_id varchar(20), foreign key (p_id) REFERENCES psngr(p_id));



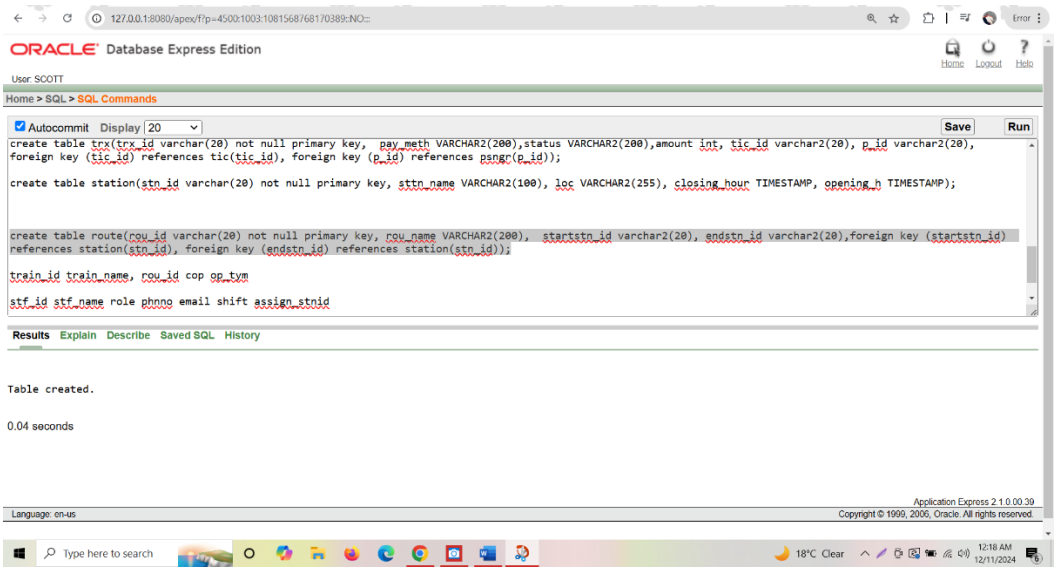
4. Transaction:
   create table trx(trx_id varchar(20) not null primary key, pay_meth VARCHAR2(200),status VARCHAR2(200),amount int, tic_id varchar2(20), p_id varchar2(20), foreign key (tic_id) references tic(tic_id), foreign key (p_id) references psngr(p_id));



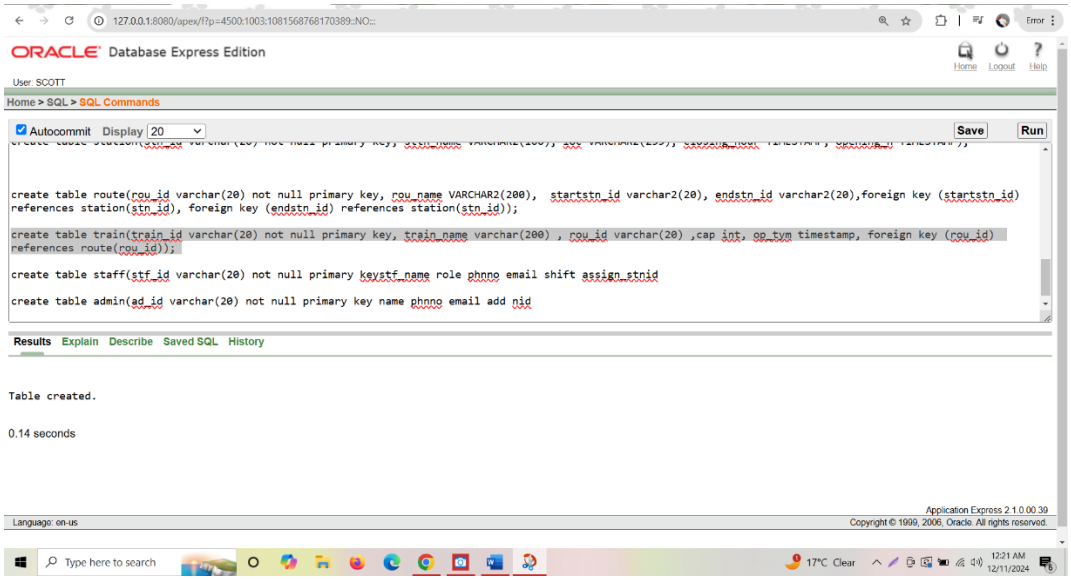
5. Station:
   create table station(stn_id varchar(20) not null primary key, sttn_name VARCHAR2(100), loc VARCHAR2(255), closing_hour TIMESTAMP, opening_h TIMESTAMP);



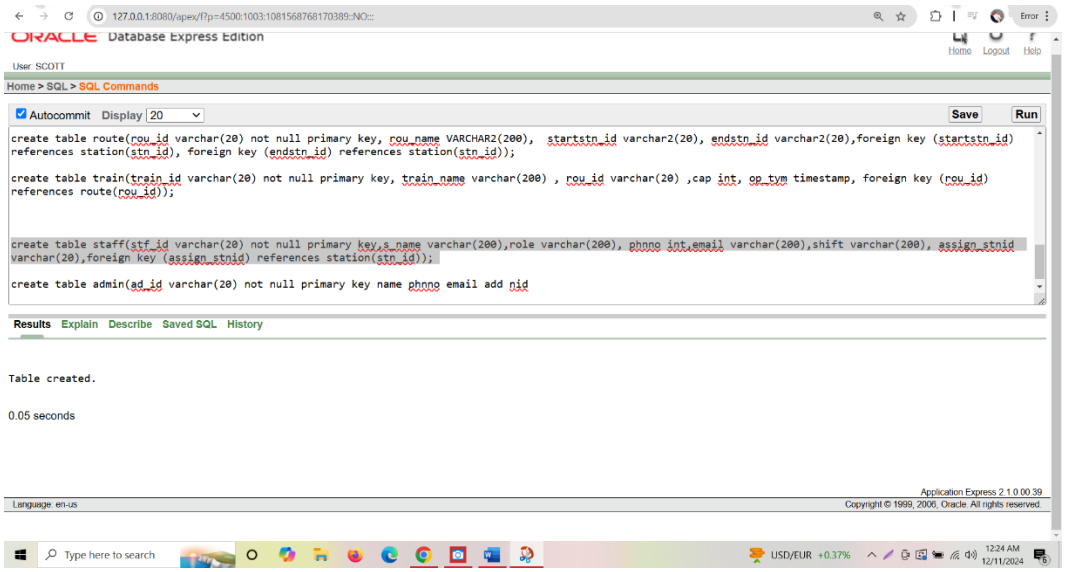
6. Route:
create table route(rou_id varchar(20) not null primary key, rou_name VARCHAR2(200), startstn_id varchar2(20), endstn_id varchar2(20),foreign key (startstn_id) references station(stn_id), foreign key (endstn_id) references station(stn_id));



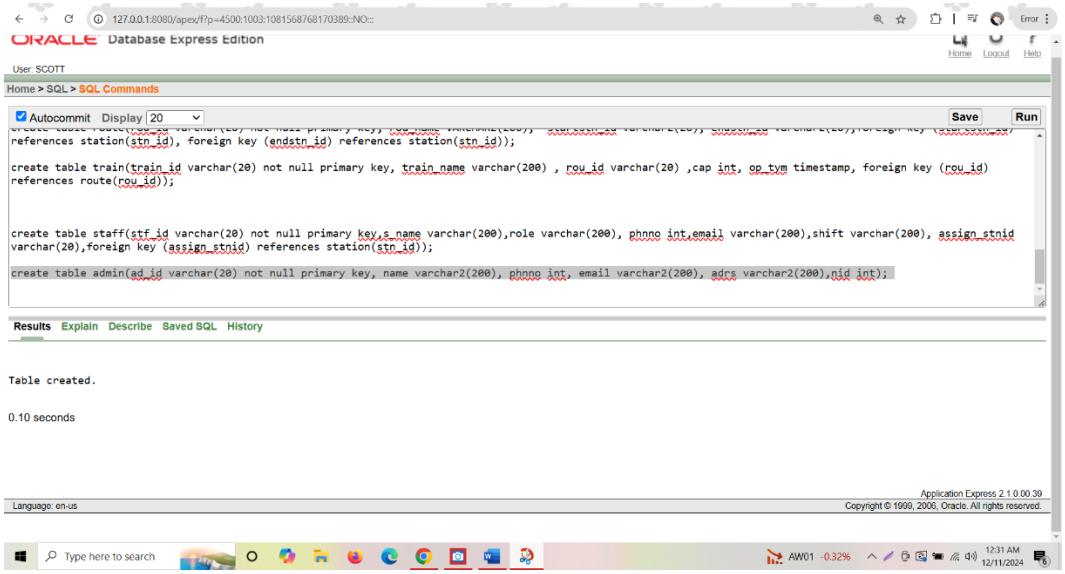
7. Train:
create table train(train_id varchar(20) not null primary key, train_name varchar(200) , rou_id varchar(20) ,cap int, op_tym varchar2(200), foreign key (rou_id) references route(rou_id));



8. Staff:
create table staff(stf_id varchar(20) not null primary key,s_name varchar(200),role varchar(200), phnno int,email varchar(200),shift varchar(200), assign_stnid varchar(20),foreign key (assign_stnid) references station(stn_id));

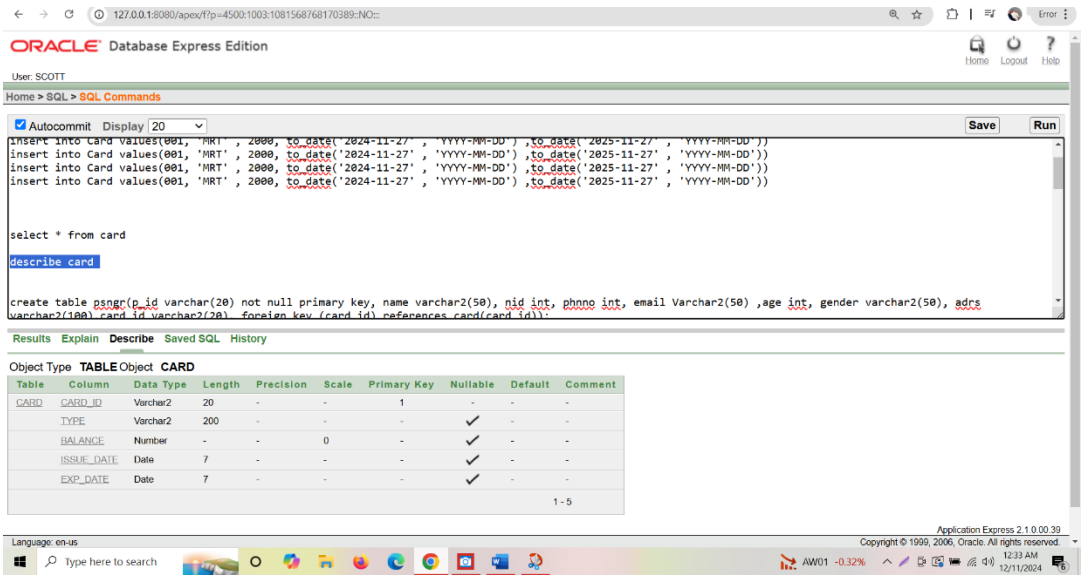


9. Admin:
create table admin(ad_id varchar(20) not null primary key, name varchar2(200), phnno int, email varchar2(200), adrs varchar2(200), nid int);

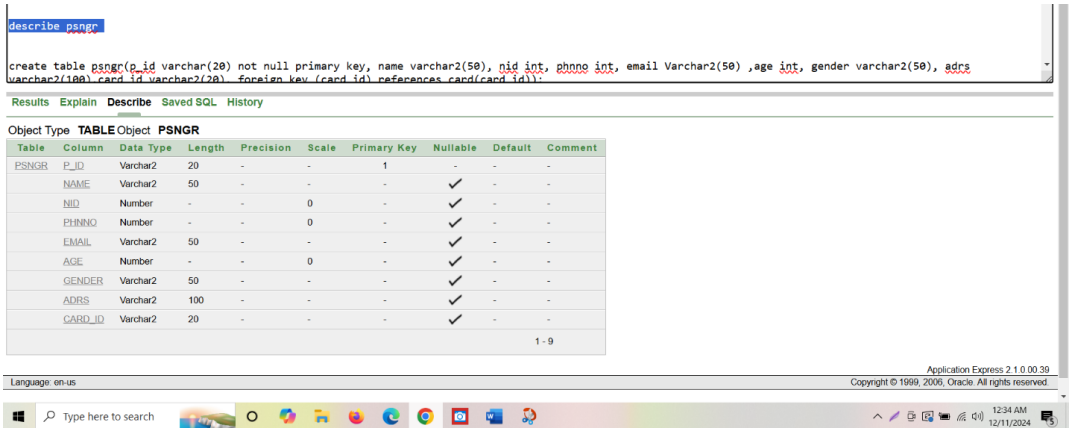


Describe Table

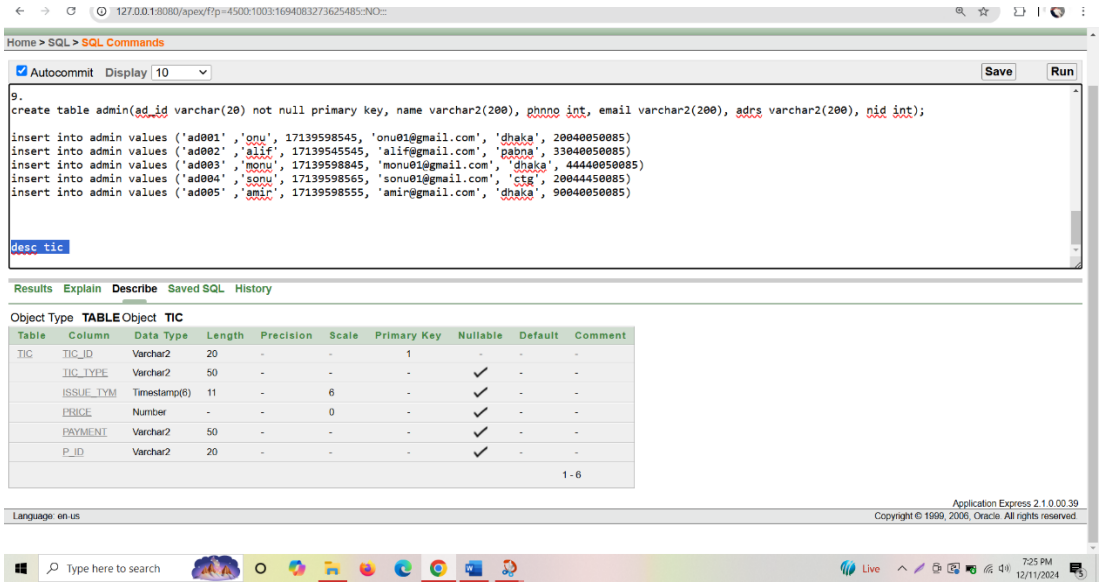
1. card



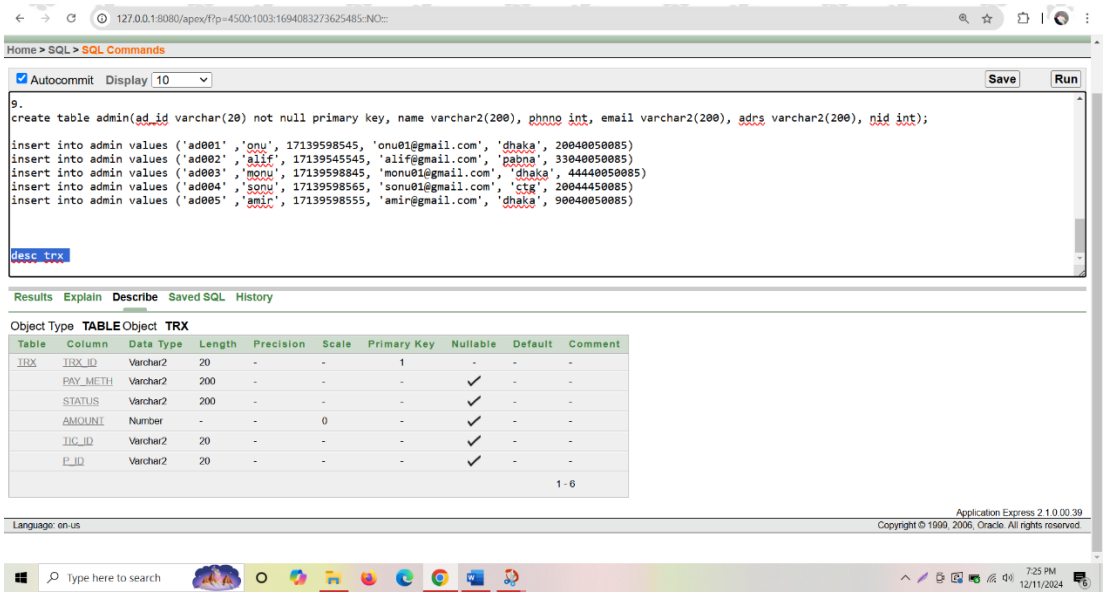
2. Psngr



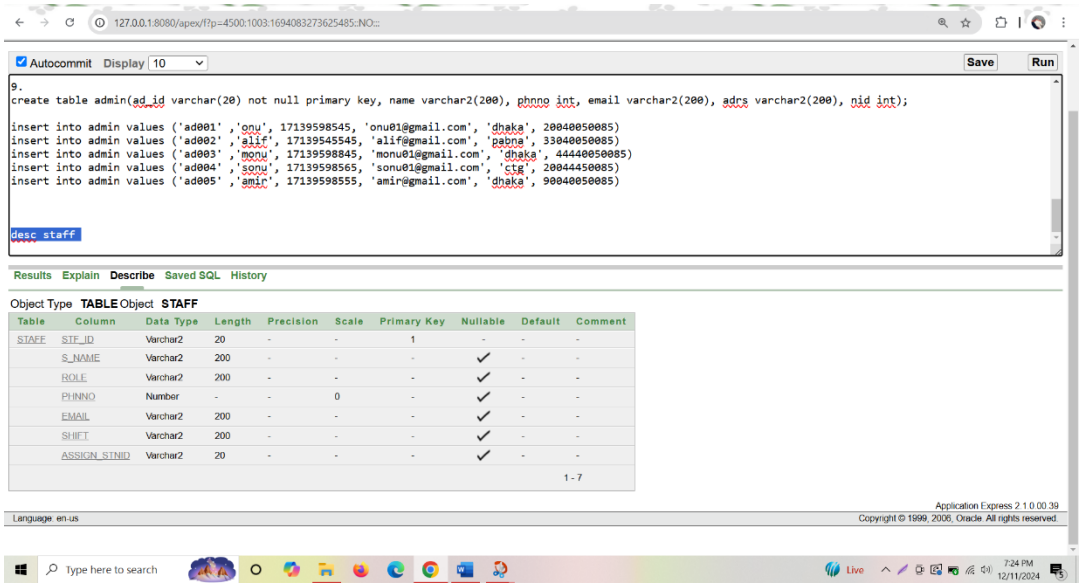
3. tic


Home > SQL > SQL Commands
Autocommit Display 10
Save
Run
9.
create table admin(ad_id varchar(20) not null primary key, name varchar2(200), phnno int, email varchar2(200), adrs varchar2(200), nid int);
insert into admin values ('ad001' ,'onu', 17139598545, 'onu01@gmail.com', 'dhaka', 20040050085)
insert into admin values ('ad002' ,'alif', 17139545545, 'alif@gmail.com', 'pabna', 33040050085)
insert into admin values ('ad003' ,'monu', 17139598845, 'monu01@gmail.com', 'dhaka', 44440050085)
insert into admin values ('ad004' ,'sonu', 17139598565, 'sonu01@gmail.com', 'ctg', 20044450085)
insert into admin values ('ad005' ,'amir', 17139598555, 'amir@gmail.com', 'dhaka', 90040050085)
desc tic
Results Explain Describe Saved SQL History
Object Type TABLE Object TIC
Table Column Data Type Length Precision Scale Primary Key Nullable Default Comment
TIC TIC_ID Varchar2 20 - - 1 - - -
TIC_TYPE Varchar2 50 - - - ✓ - -
ISSUE_TYM Timestamp(6) 11 - 6 - ✓ - -
PRICE Number - - 0 - ✓ - -
PAYMENT Varchar2 50 - - - ✓ - -
P_ID Varchar2 20 - - - ✓ - -
1 - 6
Application Express 2.1.0.00.39
Language: en-us
Copyright © 1999, 2006, Oracle. All rights reserved.
Type here to search
7:25 PM
12/11/2024

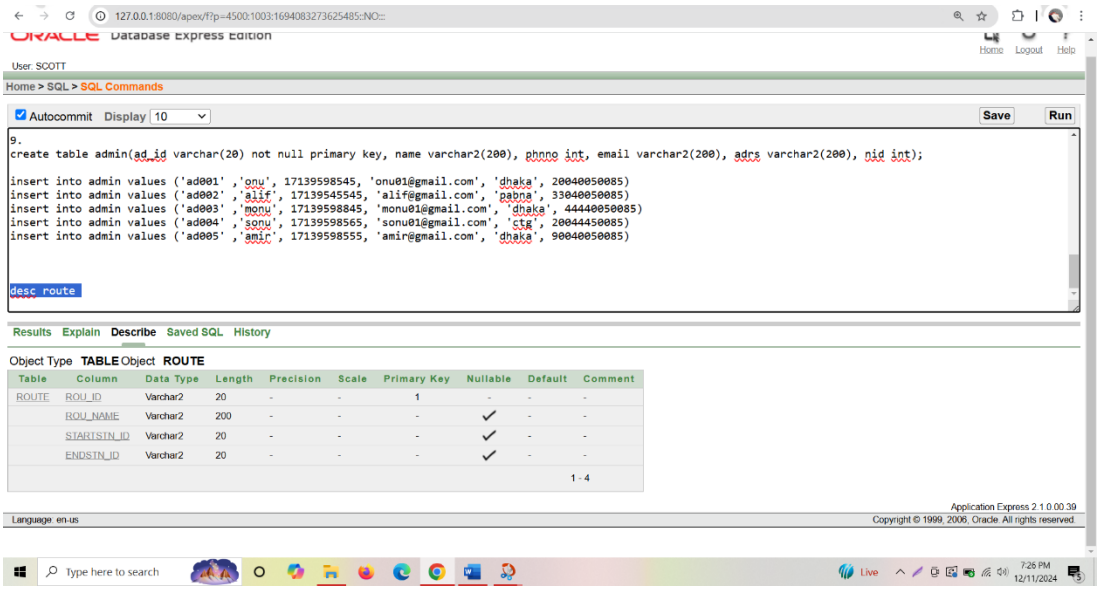
4. trx


Home > SQL > SQL Commands
Autocommit Display 10
Save
Run
9.
create table admin(ad_id varchar(20) not null primary key, name varchar2(200), phnno int, email varchar2(200), adrs varchar2(200), nid int);
insert into admin values ('ad001' ,'onu', 17139598545, 'onu01@gmail.com', 'dhaka', 20040050085)
insert into admin values ('ad002' ,'alif', 17139545545, 'alif@gmail.com', 'pabna', 33040050085)
insert into admin values ('ad003' ,'monu', 17139598845, 'monu01@gmail.com', 'dhaka', 44440050085)
insert into admin values ('ad004' ,'sonu', 17139598565, 'sonu01@gmail.com', 'ctg', 20044450085)
insert into admin values ('ad005' ,'amir', 17139598555, 'amir@gmail.com', 'dhaka', 90040050085)
desc trx
Results Explain Describe Saved SQL History
Object Type TABLE Object TRX
Table Column Data Type Length Precision Scale Primary Key Nullable Default Comment
TRX TRX_ID Varchar2 20 - - 1 - - -
PAY_METH Varchar2 200 - - - ✓ - -
STATUS Varchar2 200 - - - ✓ - -
AMOUNT Number - - 0 - ✓ - -
TIC_ID Varchar2 20 - - - ✓ - -
P_ID Varchar2 20 - - - ✓ - -
1 - 6
Application Express 2.1.0.00.39
Language: en-us
Copyright © 1999, 2006, Oracle. All rights reserved.
Type here to search
7:25 PM
12/11/2024

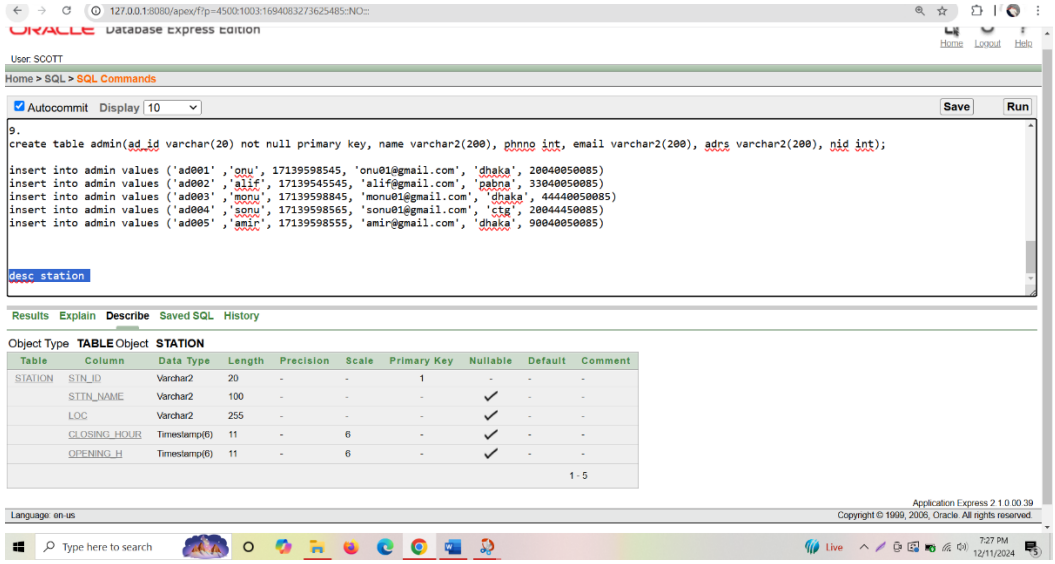
5. staff



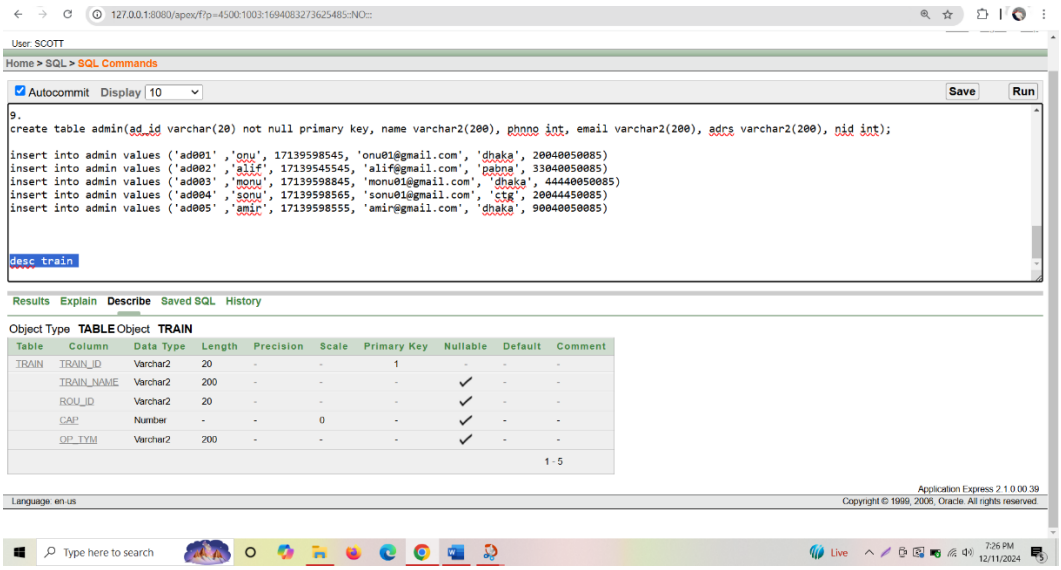
6. route



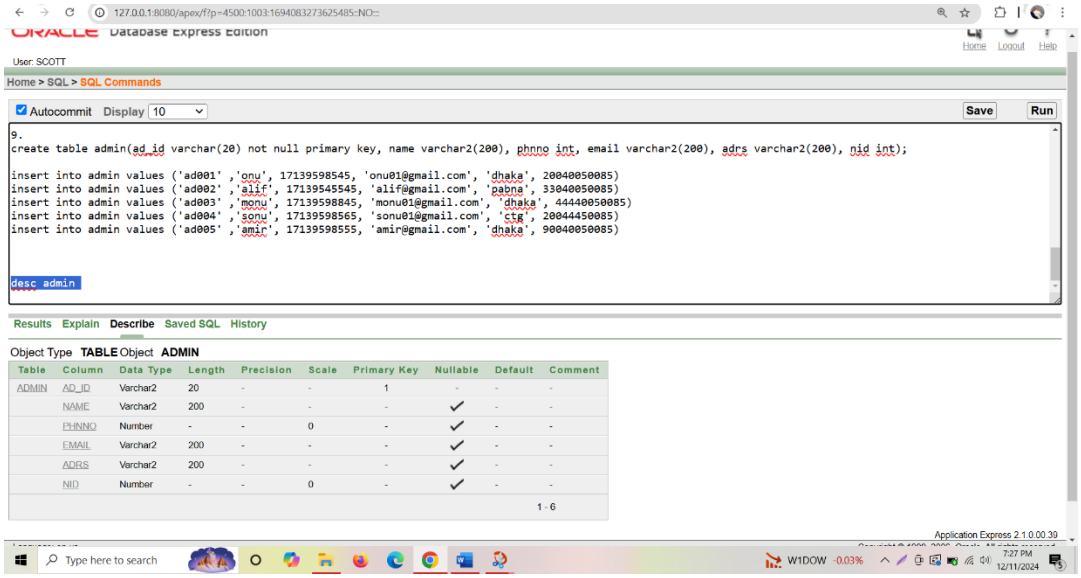
7. station



8. train

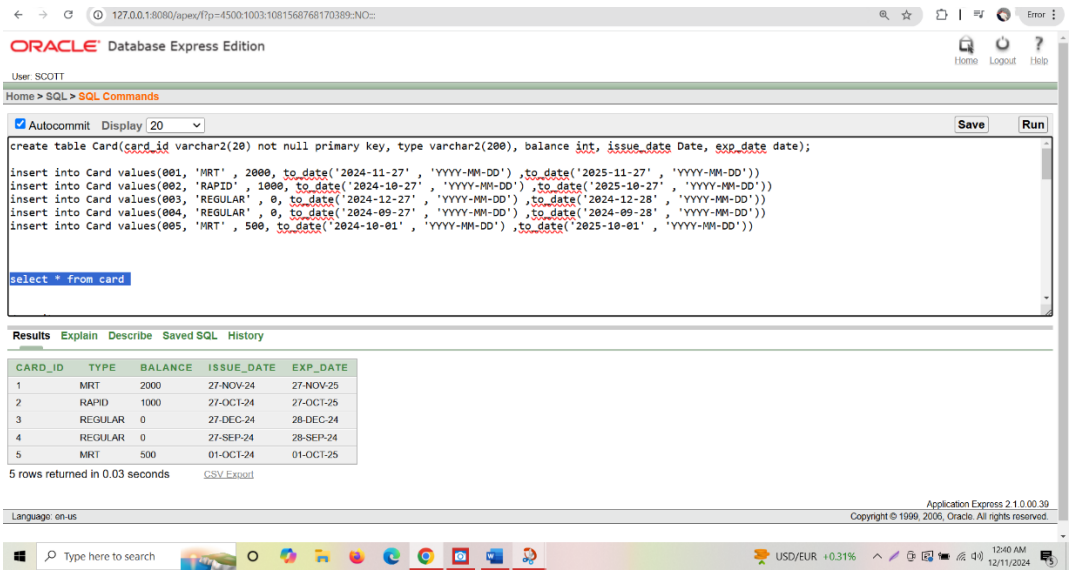


9. admin

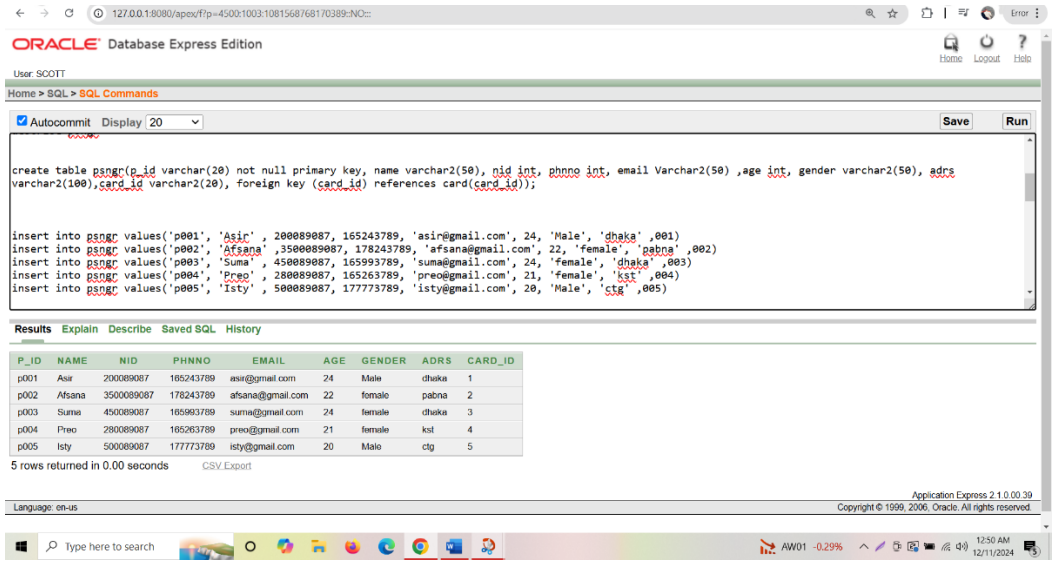


DATA INSERTION

1. CARD:
   insert into Card values(001, 'MRT' , 2000, to_date('2024-11-27' , 'YYYY-MM-DD') ,to_date('2025-11-27' , 'YYYY-MM-DD'))
   insert into Card values(002, 'RAPID' , 1000, to_date('2024-10-27' , 'YYYY-MM-DD') ,to_date('2025-10-27' , 'YYYY-MM-DD'))
   insert into Card values(003, 'REGULAR' , 0, to_date('2024-12-27' , 'YYYY-MM-DD') ,to_date('2024-12-28' , 'YYYY-MM-DD'))
   insert into Card values(004, 'REGULAR' , 0, to_date('2024-09-27' , 'YYYY-MM-DD') ,to_date('2024-09-28' , 'YYYY-MM-DD'))
   insert into Card values(005, 'MRT' , 500, to_date('2024-10-01' , 'YYYY-MM-DD') ,to_date('2025-10-01' , 'YYYY-MM-DD'))

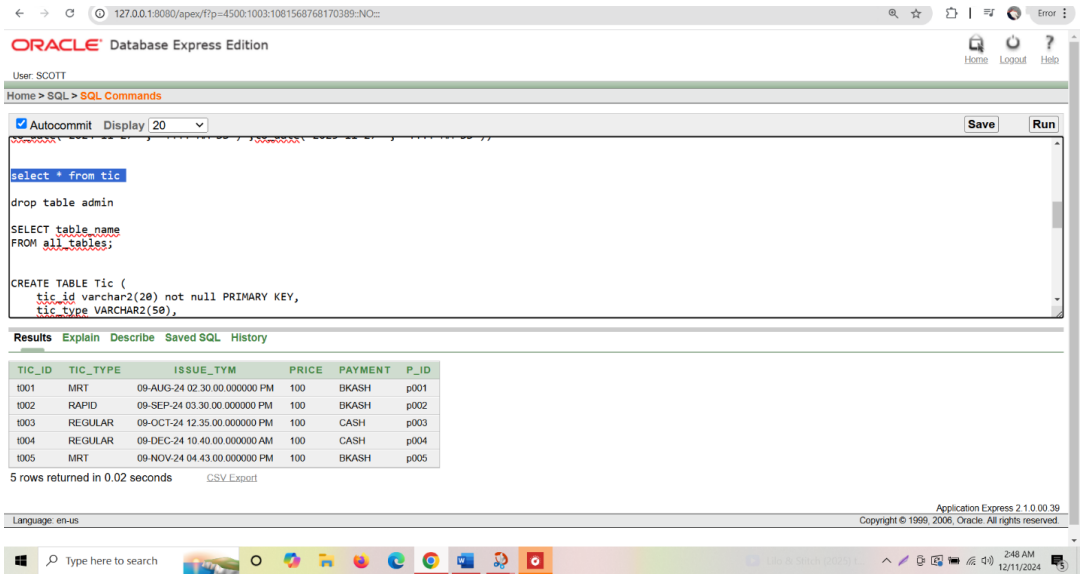


2. Psngr:
insert into psngr values('p001', 'Asir' , 200089087, 165243789, 'asir@gmail.com', 24, 'Male', 'dhaka' ,001)
insert into psngr values('p002', 'Afsana' ,3500089087, 178243789, 'afsana@gmail.com', 22, 'female', 'pabna' ,002)
insert into psngr values('p003', 'Suma' , 450089087, 165993789, 'suma@gmail.com', 24, 'female', 'dhaka' ,003)
insert into psngr values('p004', 'Preo' , 280089087, 165263789, 'preo@gmail.com', 21, 'female', 'kst' ,004)
insert into psngr values('p005', 'Isty' , 500089087, 177773789, 'isty@gmail.com', 20, 'Male', 'ctg' ,005)



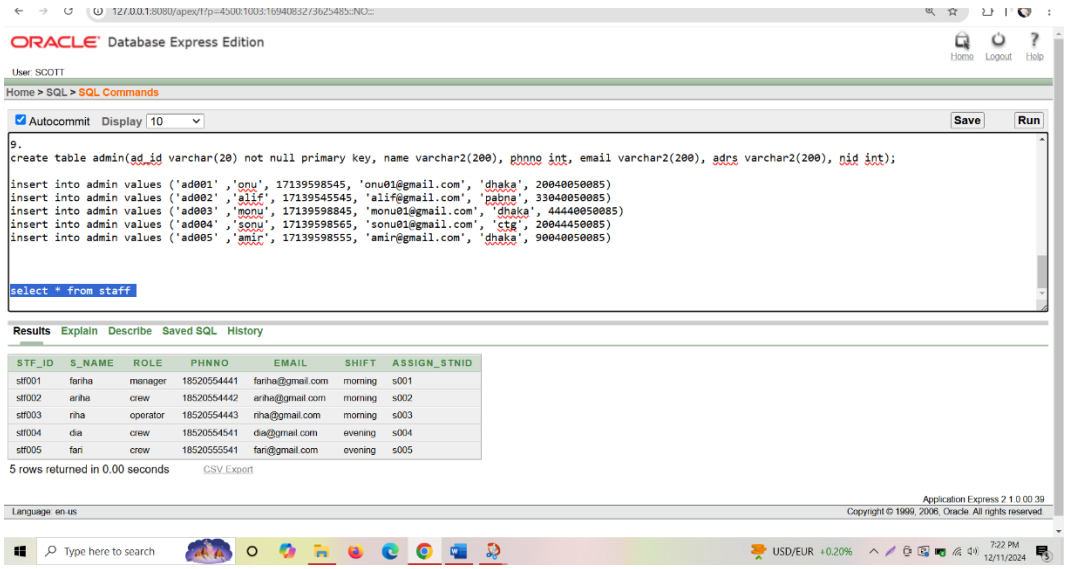
3. TIC:

insert into TIC values('t001', 'MRT' , TIMESTAMP '2024-08-09 14:30:00',100, 'BKASH' ,'p001')
insert into tic values('t002', 'RAPID' ,TIMESTAMP '2024-09-09 15:30:00',100, 'BKASH' ,'p002')
insert into tic values('t003', 'REGULAR' , TIMESTAMP '2024-10-09 12:35:00',100, 'CASH' ,'p003')
insert into tic values('t004', 'REGULAR' , TIMESTAMP '2024-12-09 10:40:00',100, 'CASH' ,'p004')
insert into tic values('t005', 'MRT' , TIMESTAMP '2024-11-09 16:43:00',100, 'BKASH' ,'p005')



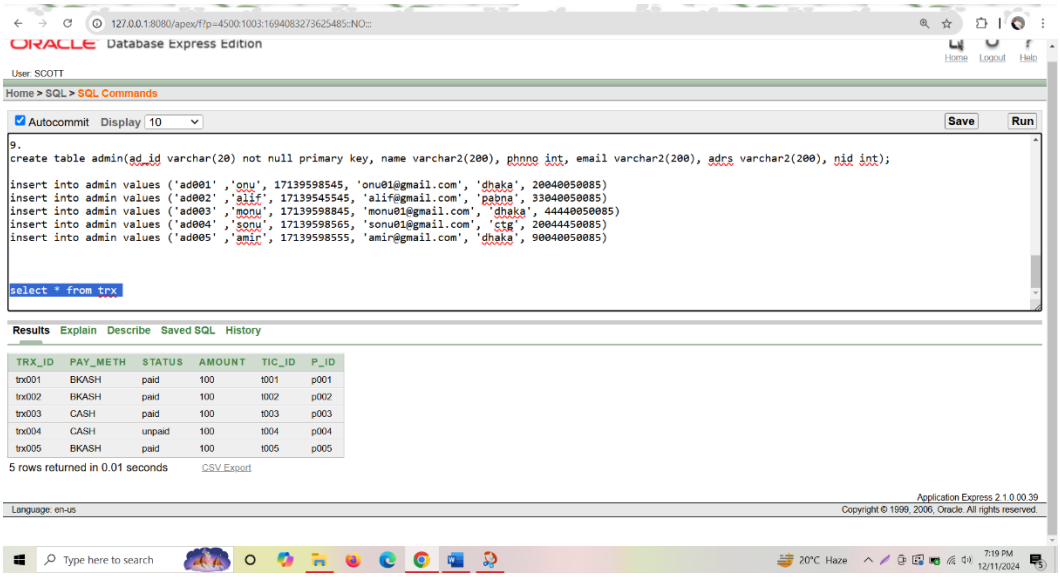
4. Staff:

insert into staff values ('stf001' , 'fariha', 'manager' ,18520554441 ,'fariha@gmail.com' , 'morning' , 's001')
insert into staff values ('stf002' , 'ariha', 'crew' ,18520554442 ,'ariha@gmail.com' , 'morning' , 's002')
insert into staff values ('stf003' , 'riha', 'operator' ,18520554443 ,'riha@gmail.com' , 'morning' , 's003')
insert into staff values ('stf004' , 'dia', 'crew' ,18520554541 ,'dia@gmail.com' , 'evening' , 's004')
insert into staff values ('stf005' , 'fari', 'crew' ,18520555541 ,'fari@gmail.com' , 'evening' , 's005')

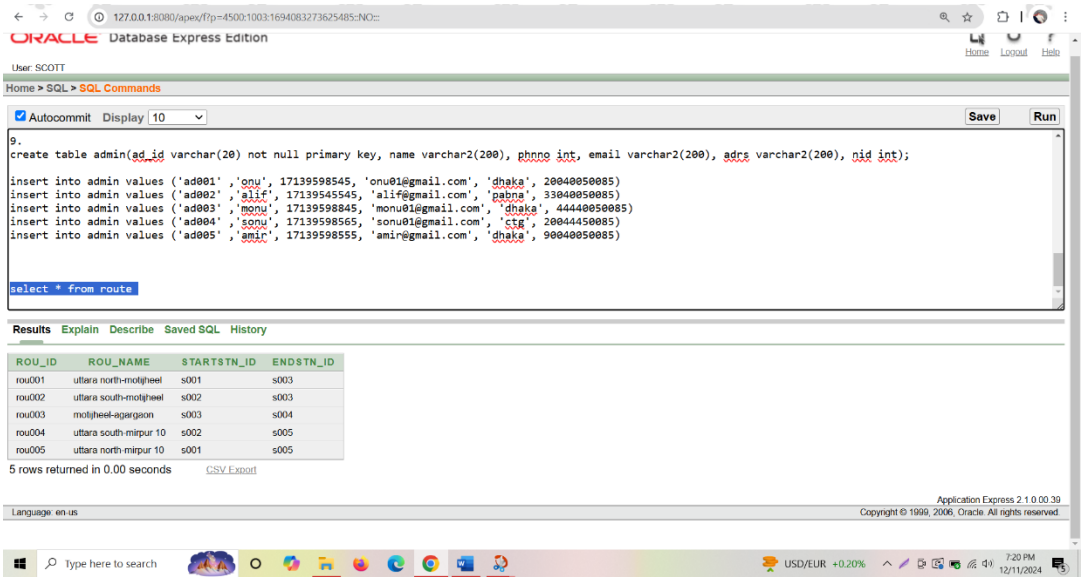


5. Trx:

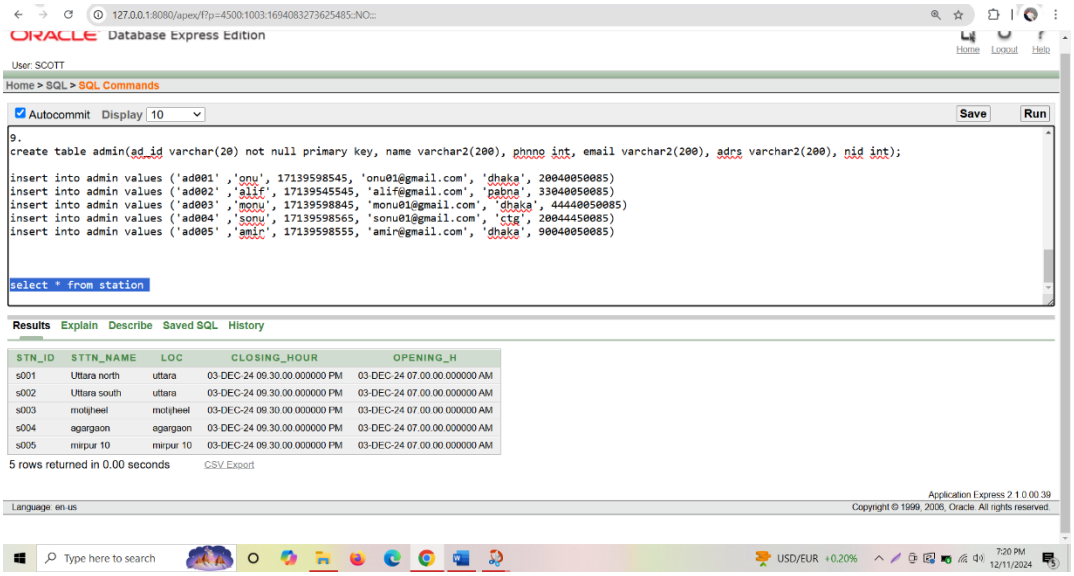
insert into trx values('trx001' , 'BKASH' , 'paid' , 100, 't001' , 'p001')
insert into trx values('trx002' , 'BKASH' , 'paid' , 100, 't002' , 'p002')
insert into trx values('trx003' , 'CASH' , 'paid' , 100, 't003' , 'p003')
insert into trx values('trx004' , 'CASH' , 'unpaid' , 100, 't004' , 'p004')
insert into trx values('trx005' , 'BKASH' , 'paid' , 100, 't005' , 'p005')



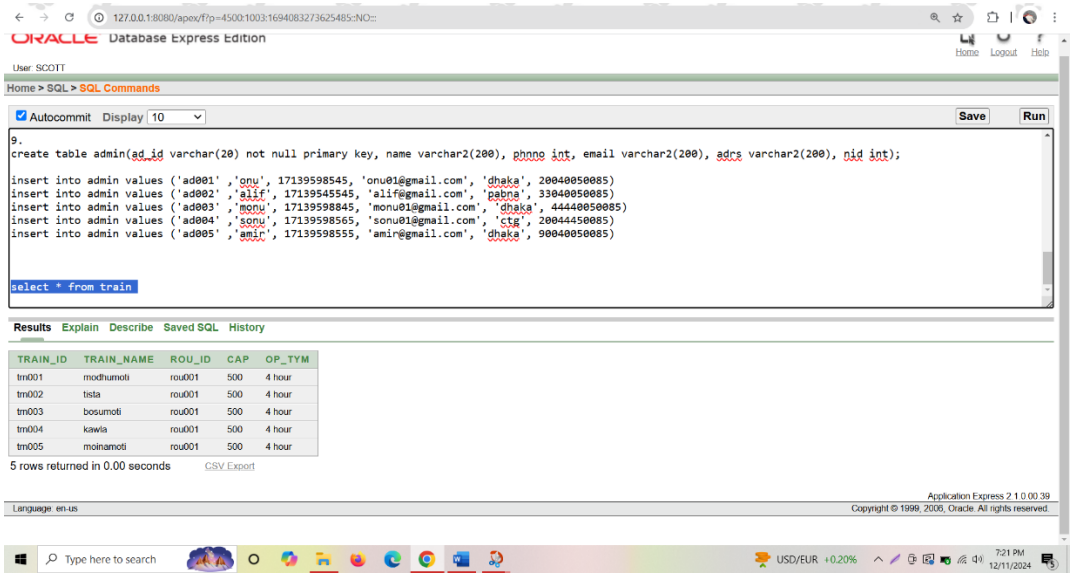
6. Route:
   insert into route values ('rou001','uttara north-motijheel' , 's001' ,'s003')
   insert into route values ('rou002','uttara south-motijheel' , 's002' ,'s003')
   insert into route values ('rou003','motijheel-agargaon' , 's003' ,'s004')
   insert into route values ('rou004','uttara south-mirpur 10' , 's002' ,'s005')
   insert into route values ('rou005','uttara north-mirpur 10' , 's001' ,'s005')



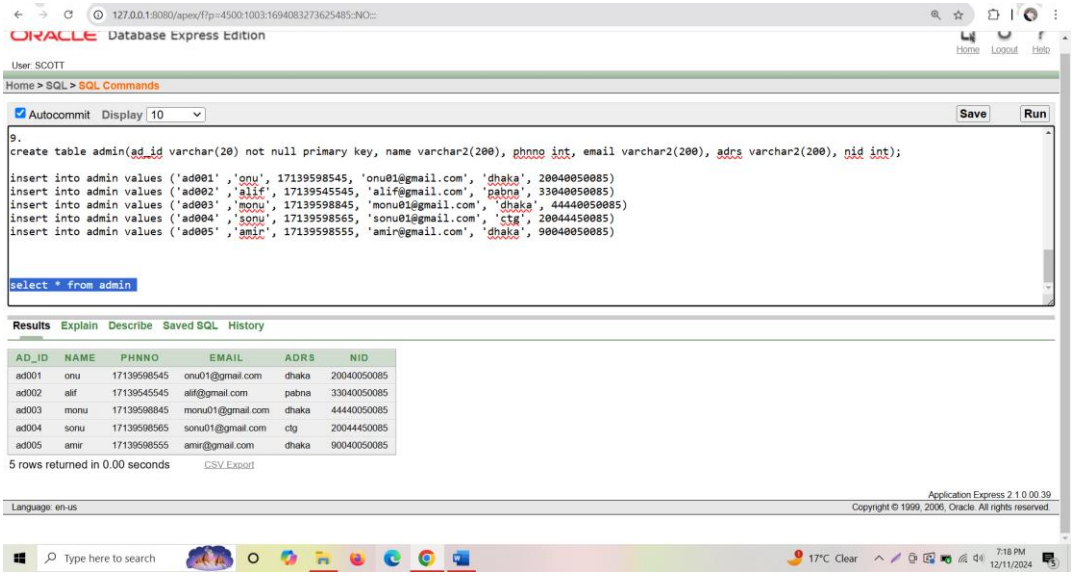
7. Station:
   insert into station values ('s001' , 'Uttara north' ,'uttara' , timestamp '2024-12-03 21:30:00' ,timestamp '2024-12-03 07:00:00')
   insert into station values ('s002' , 'Uttara south' ,'uttara' , timestamp ' 2024-12-03 21:30:00' ,timestamp '2024-12-03 07:00:00')
   insert into station values ('s003' , 'motijheel' ,'motijheel' , timestamp ' 2024-12-03 21:30:00' ,timestamp '2024-12-03 07:00:00')
   insert into station values ('s004' , 'agargaon' ,'agargaon' , timestamp ' 2024-12-03 21:30:00' ,timestamp '2024-12-03 07:00:00')
   insert into station values ('s005' , 'mirpur 10' ,'mirpur 10' , timestamp ' 2024-12-03 21:30:00' ,timestamp '2024-12-03 07:00:00')



8. Train:
insert into train values ('trn001' , 'modhumoti', 'rou001' ,500, '4 hour')
insert into train values ('trn002' , 'tista', 'rou001' ,500, '4 hour')
insert into train values ('trn003' , 'bosumoti', 'rou001' ,500, '4 hour')
insert into train values ('trn004' , 'kawla', 'rou001' ,500, '4 hour')
insert into train values ('trn005' , 'moinamoti', 'rou001' ,500, '4 hour')



9. Admin:
   insert into admin values ('ad001' ,'onu', 17139598545, 'onu01@gmail.com', 'dhaka', 20040050085)
   insert into admin values ('ad002' ,'alif', 17139545545, 'alif@gmail.com', 'pabna', 33040050085)
   insert into admin values ('ad003' ,'monu', 17139598845, 'monu01@gmail.com', 'dhaka', 44440050085)
   insert into admin values ('ad004' ,'sonu', 17139598565, 'sonu01@gmail.com', 'ctg', 20044450085)
   insert into admin values ('ad005' ,'amir', 17139598555, 'amir@gmail.com', 'dhaka', 90040050085)



Basic PL/SQL

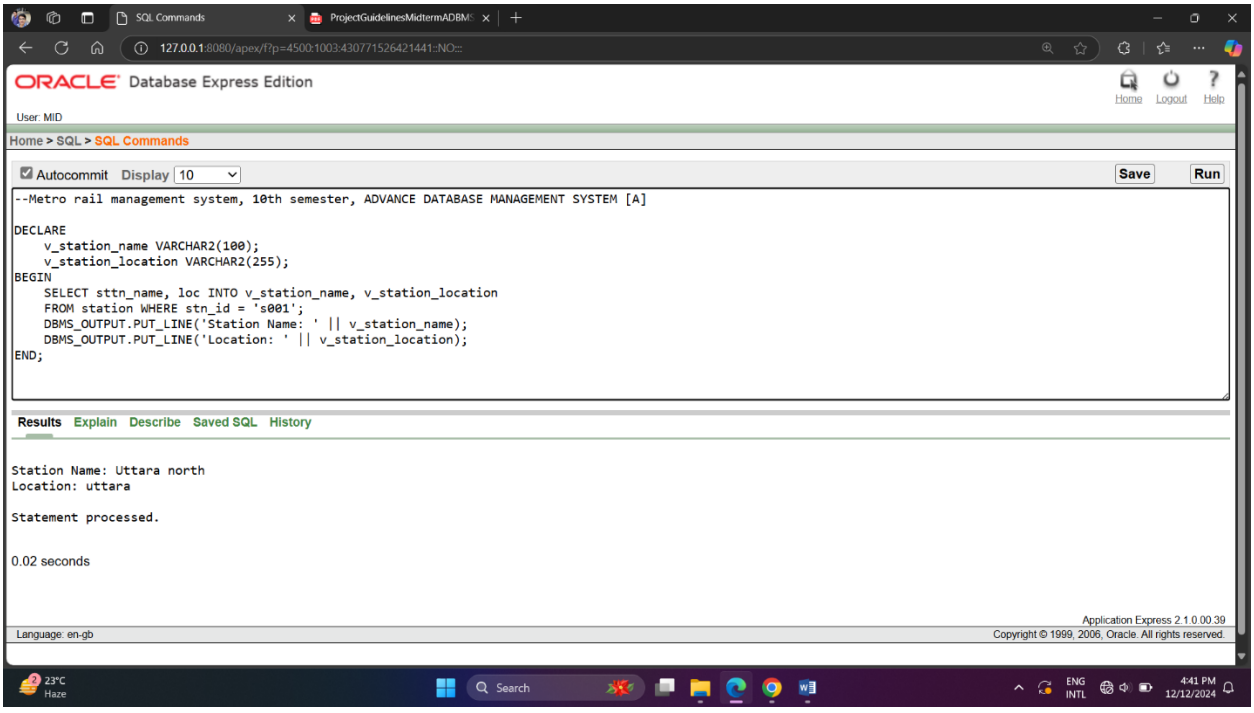
variables

a) Write a query that can display Station_name and location using pl/sql

Ans:

```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

DECLARE
   v_station_name VARCHAR2(100);
   v_station_location VARCHAR2(255);
BEGIN
   SELECT sttn_name, loc INTO v_station_name, v_station_location
   FROM station WHERE stn_id = 's001';
   DBMS_OUTPUT.PUT_LINE('Station Name: ' || v_station_name);
   DBMS_OUTPUT.PUT_LINE('Location: ' || v_station_location);
END;
```

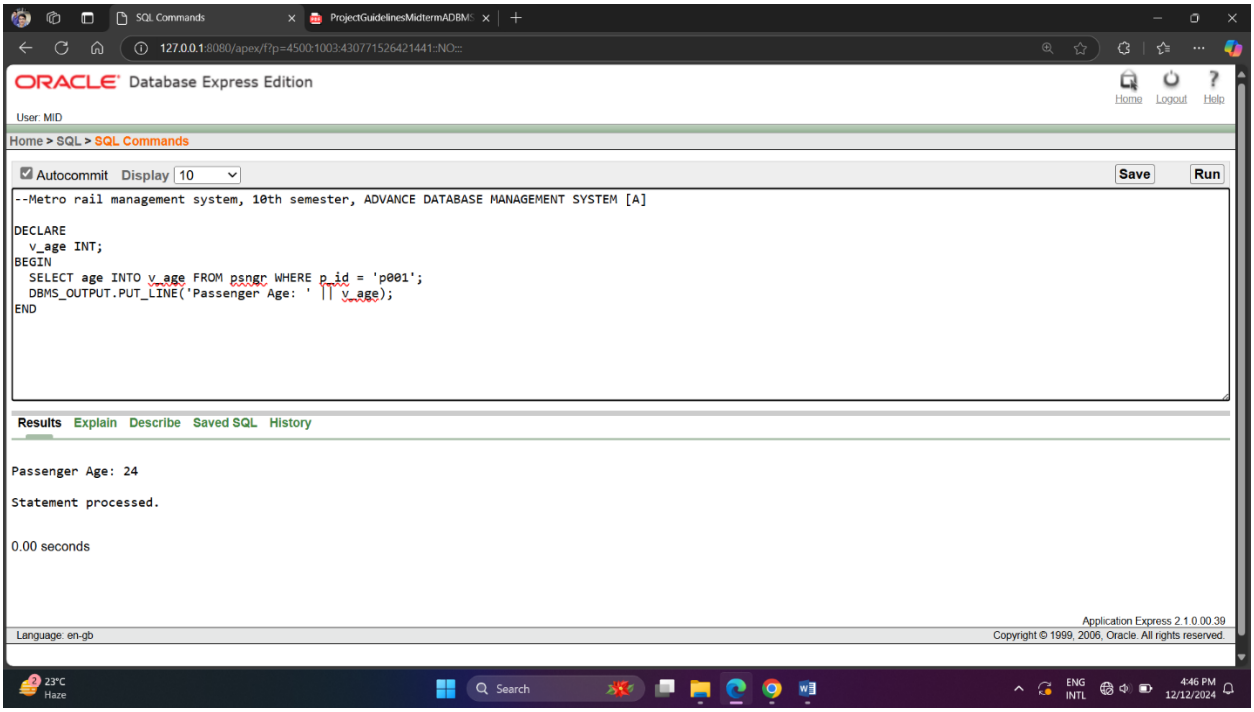


b) Write a query that can display passenger age from psngr table where p_id p001 using pl/sql

Ans:

```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM
[A]

DECLARE
 v_age INT;
BEGIN
 SELECT age INTO v_age FROM psngr WHERE p_id = 'p001';
 DBMS_OUTPUT.PUT_LINE('Passenger Age: ' || v_age);
END
```



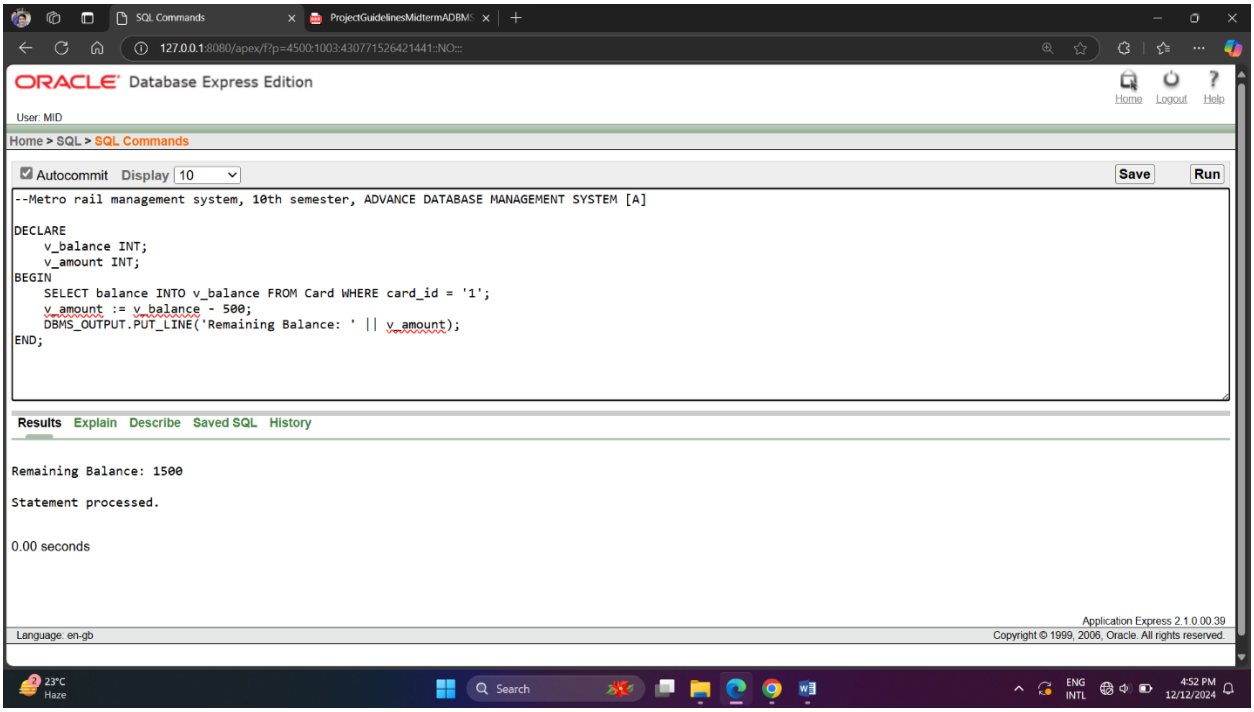
operators

a) Write a query that display remaining balance after removing 500 from card where id = 1

Ans:

```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM
[A]

DECLARE
   v_balance INT;
   v_amount INT;
BEGIN
   SELECT balance INTO v_balance FROM Card WHERE card_id = '1';
   v_amount := v_balance - 500;
   DBMS_OUTPUT.PUT_LINE('Remaining Balance: ' || v_amount);
END;
```

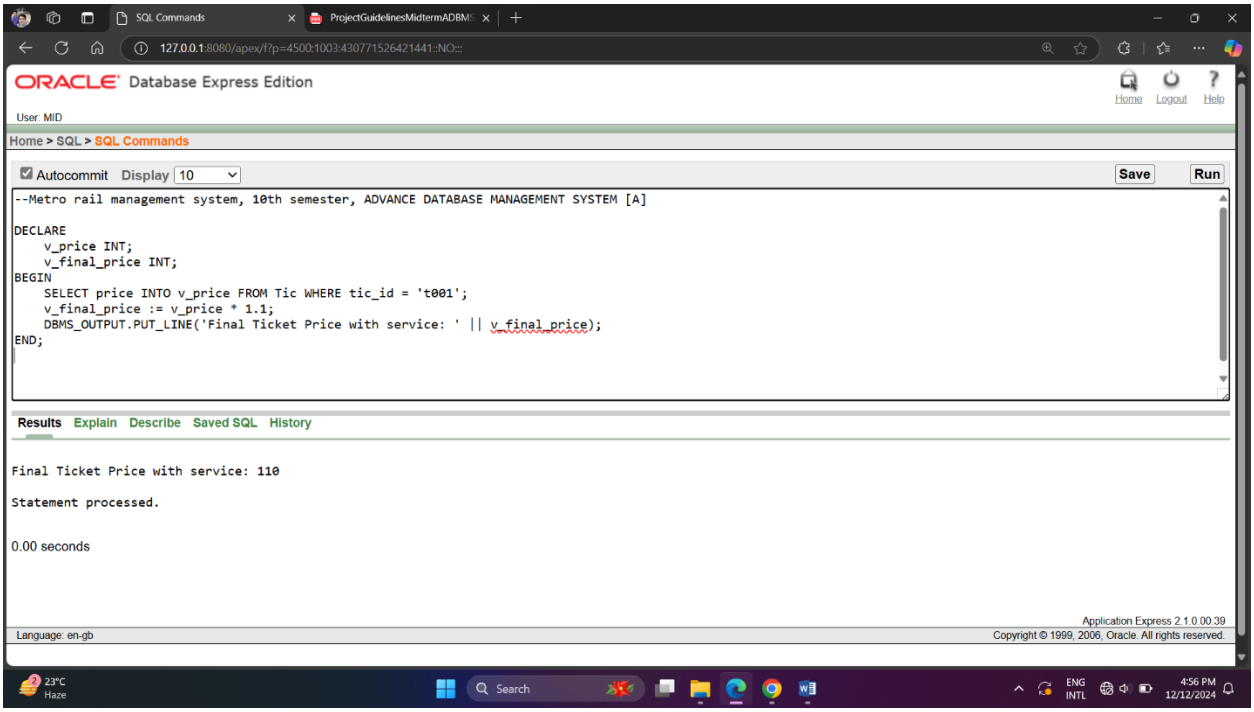


b) Write a query that display ticket price including charges where ticket id t001

Ans:

```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

DECLARE
   v_price INT;
   v_final_price INT;
BEGIN
   SELECT price INTO v_price FROM Tic WHERE tic_id = 't001';
   v_final_price := v_price * 1.1;
   DBMS_OUTPUT.PUT_LINE('Final Ticket Price with service: ' || v_final_price);
END;
```



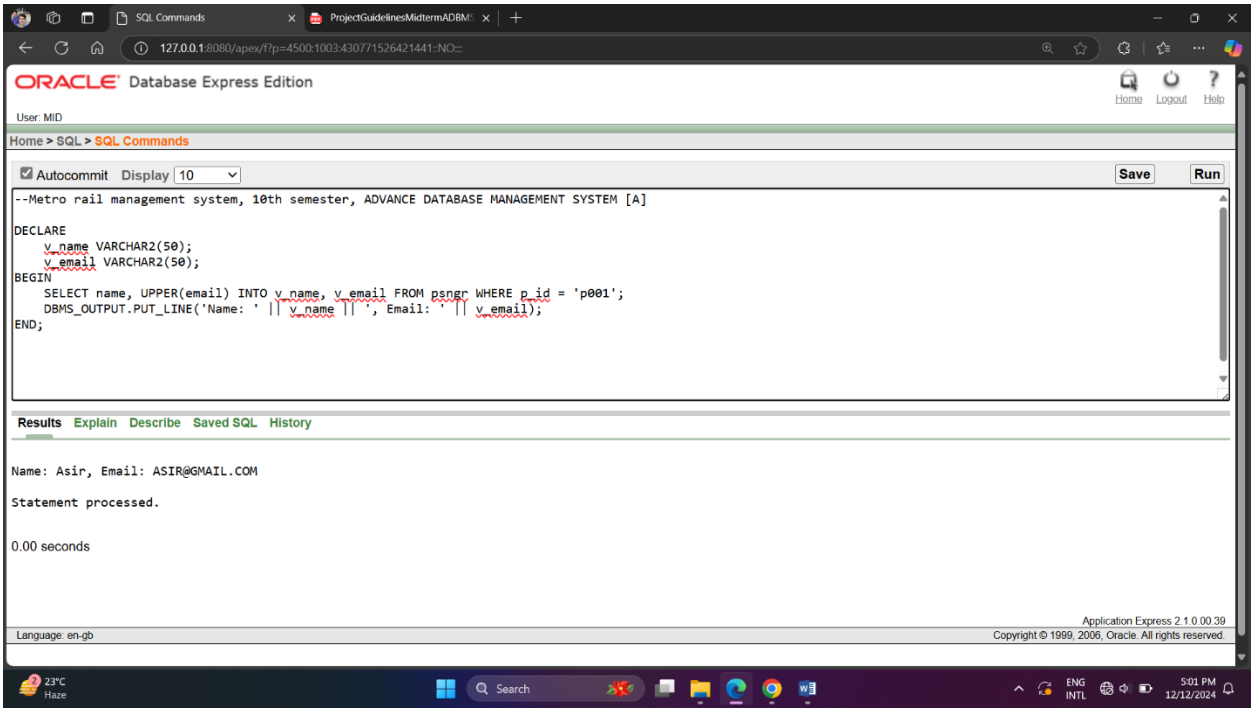
single-row function

a) Write a query that display name and email from psngr table where email show all in capital letter.

Ans:

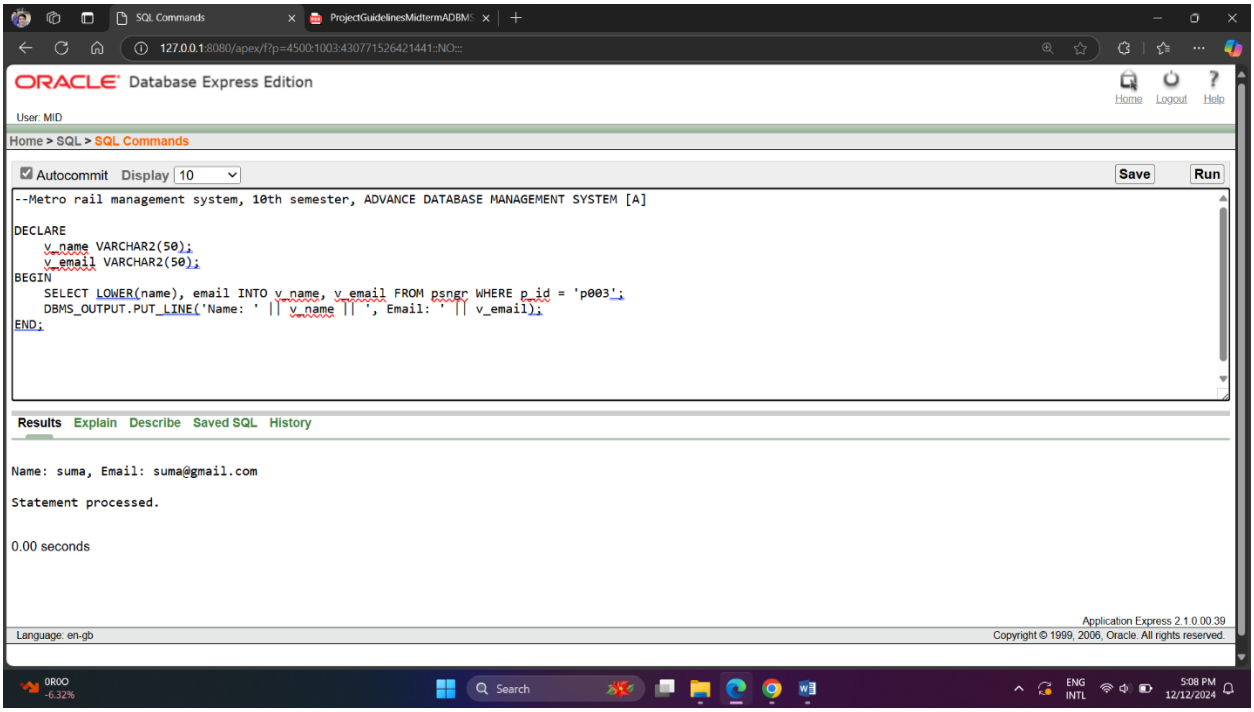
```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

DECLARE
   v_name VARCHAR2(50);
   v_email VARCHAR2(50);
BEGIN
   SELECT name, UPPER(email) INTO v_name, v_email FROM psngr WHERE p_id = 'p001';
   DBMS_OUTPUT.PUT_LINE('Name: ' || v_name || ', Email: ' || v_email);
END;
```



b) Write a query that display name and email from psngr table where name show all in small letter.
Ans: --Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

```
DECLARE
  v_name VARCHAR2(50);
  v_email VARCHAR2(50);
BEGIN
  SELECT LOWER(name), email INTO v_name, v_email FROM psngr WHERE p_id = 'p003';
  DBMS_OUTPUT.PUT_LINE('Name: ' || v_name || ', Email: ' || v_email);
END;
```



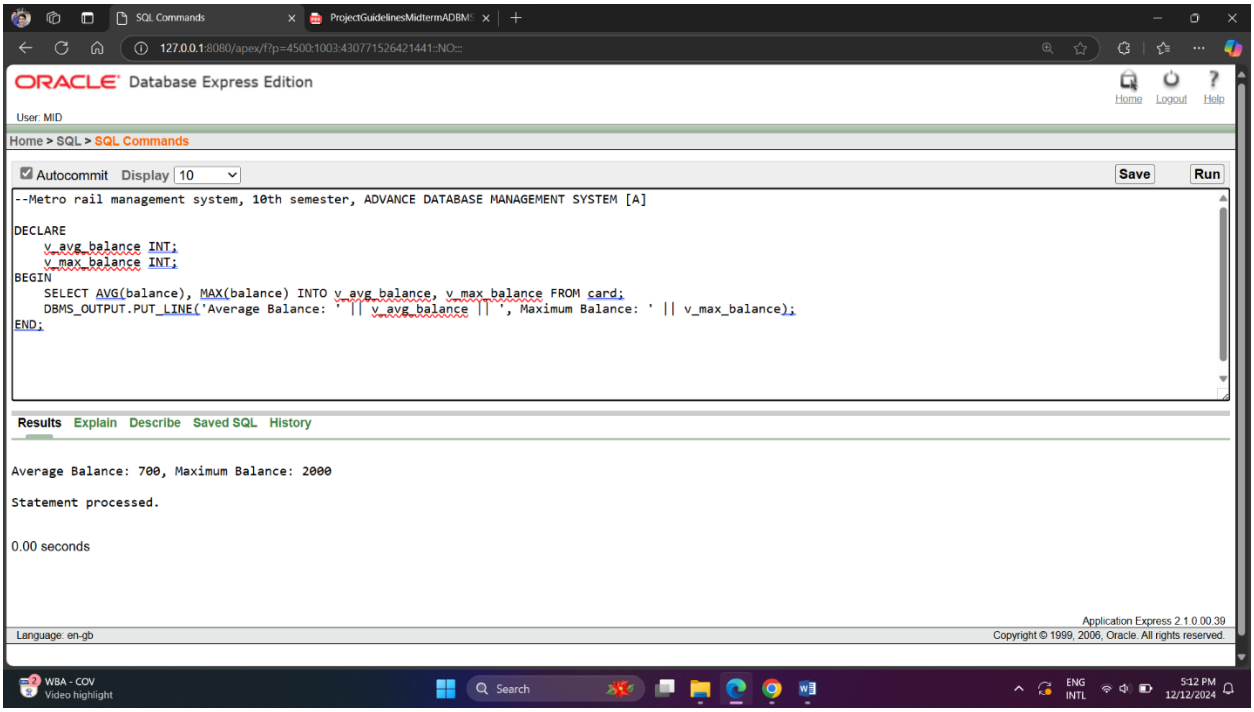
Group function

a) Write a query that display average and maximum balance from card table

Ans:

--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

```
DECLARE
   v_avg_balance INT;
   v_max_balance INT;
BEGIN
   SELECT AVG(balance), MAX(balance) INTO v_avg_balance, v_max_balance FROM card;
   DBMS_OUTPUT.PUT_LINE('Average Balance: ' || v_avg_balance || ', Maximum Balance: ' ||
v_max_balance);
END;
```

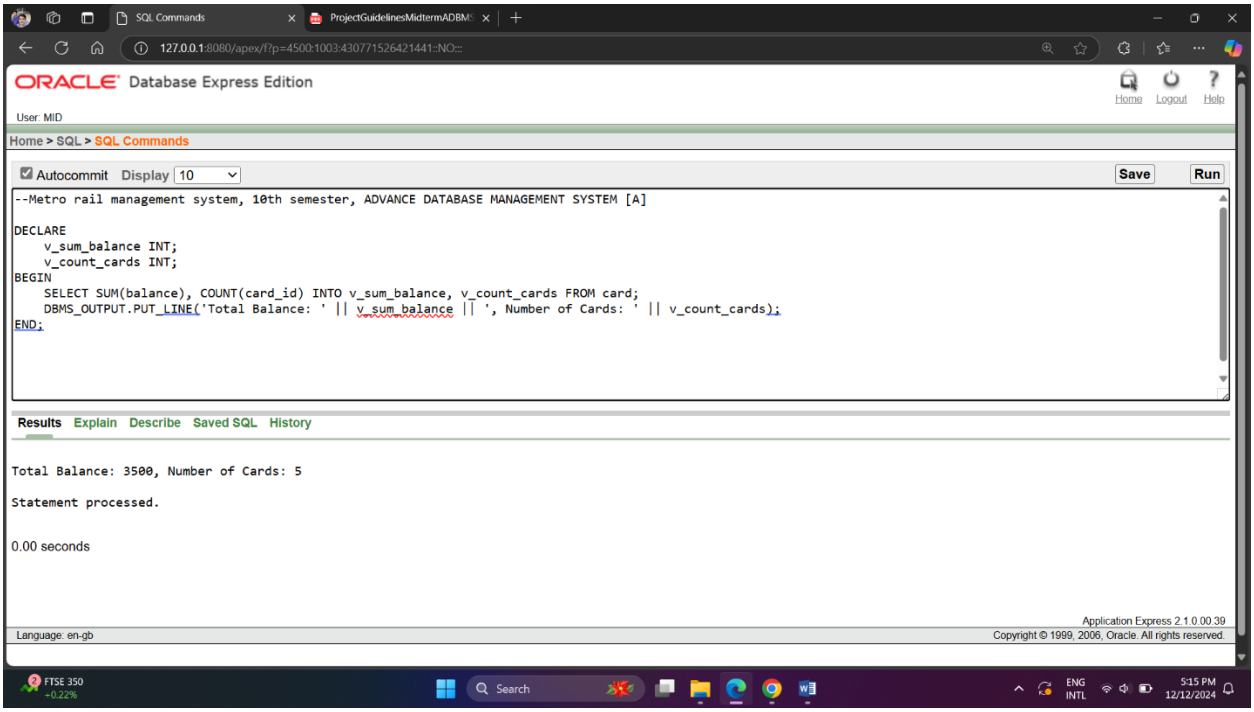


b) Write a query that display total balance and total card number from card table

Ans:

--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

```
DECLARE
   v_sum_balance INT;
   v_count_cards INT;
BEGIN
   SELECT SUM(balance), COUNT(card_id) INTO v_sum_balance, v_count_cards FROM card;
   DBMS_OUTPUT.PUT_LINE('Total Balance: ' || v_sum_balance || ', Number of Cards: ' ||
v_count_cards);
END;
```



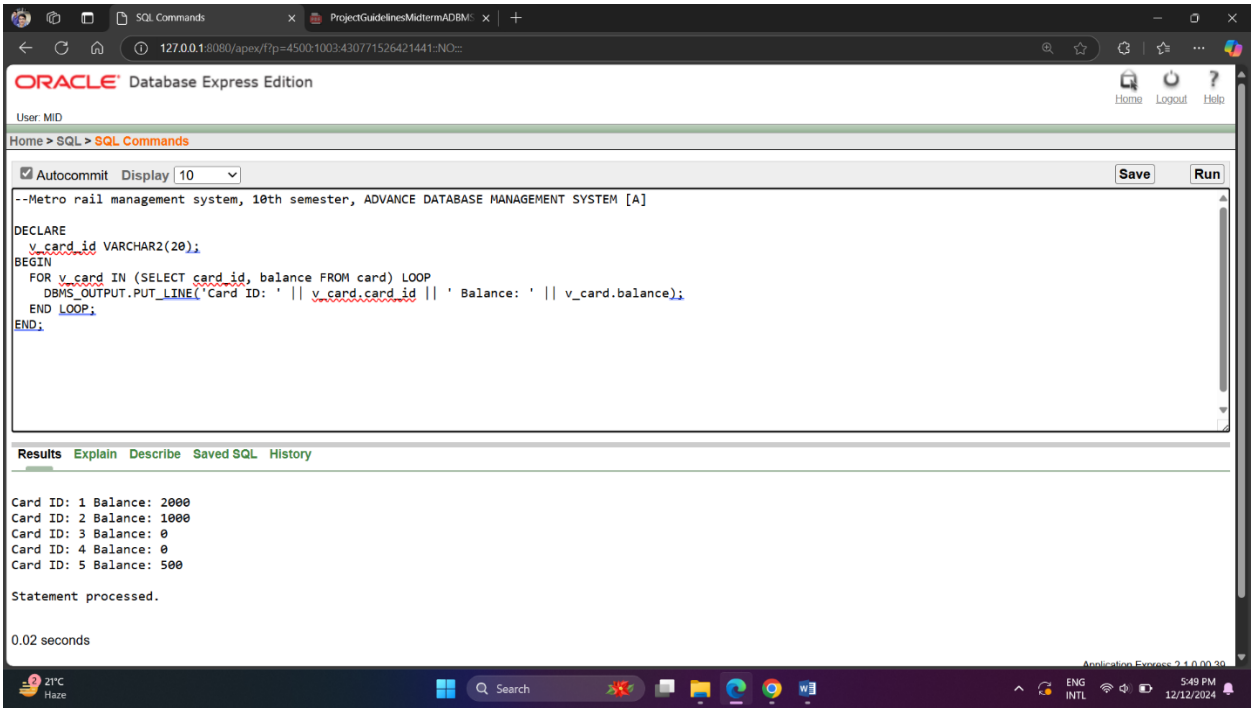
Loop

a) Write a query that display card balance using for loop

Ans:

```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM
[A]

DECLARE
 v_card_id VARCHAR2(20);
BEGIN
 FOR v_card IN (SELECT card_id, balance FROM card) LOOP
  DBMS_OUTPUT.PUT_LINE('Card ID: ' || v_card.card_id || ' Balance: ' || v_card.balance);
 END LOOP;
END;
```

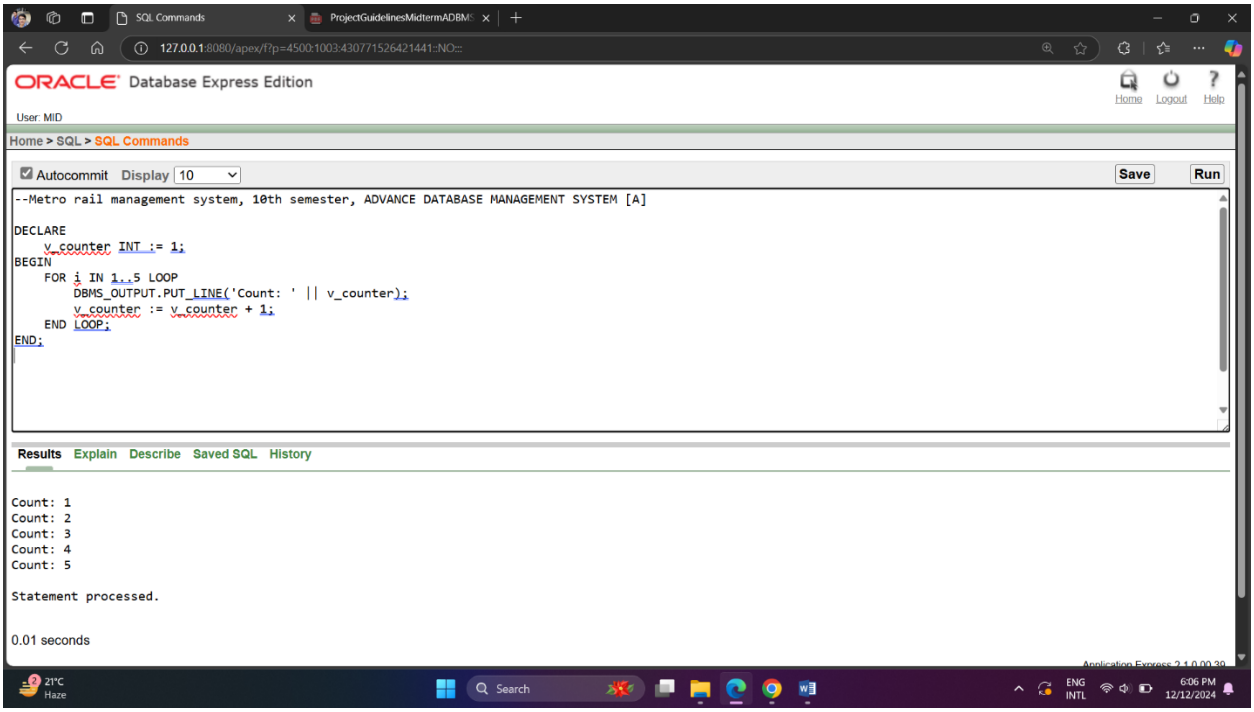


b) Write a query that display number as counter using loop

Ans:

--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

```
DECLARE
  v_counter INT := 1;
BEGIN
  FOR i IN 1..5 LOOP
    DBMS_OUTPUT.PUT_LINE('Count: ' || v_counter);
    v_counter := v_counter + 1;
  END LOOP;
END;
```



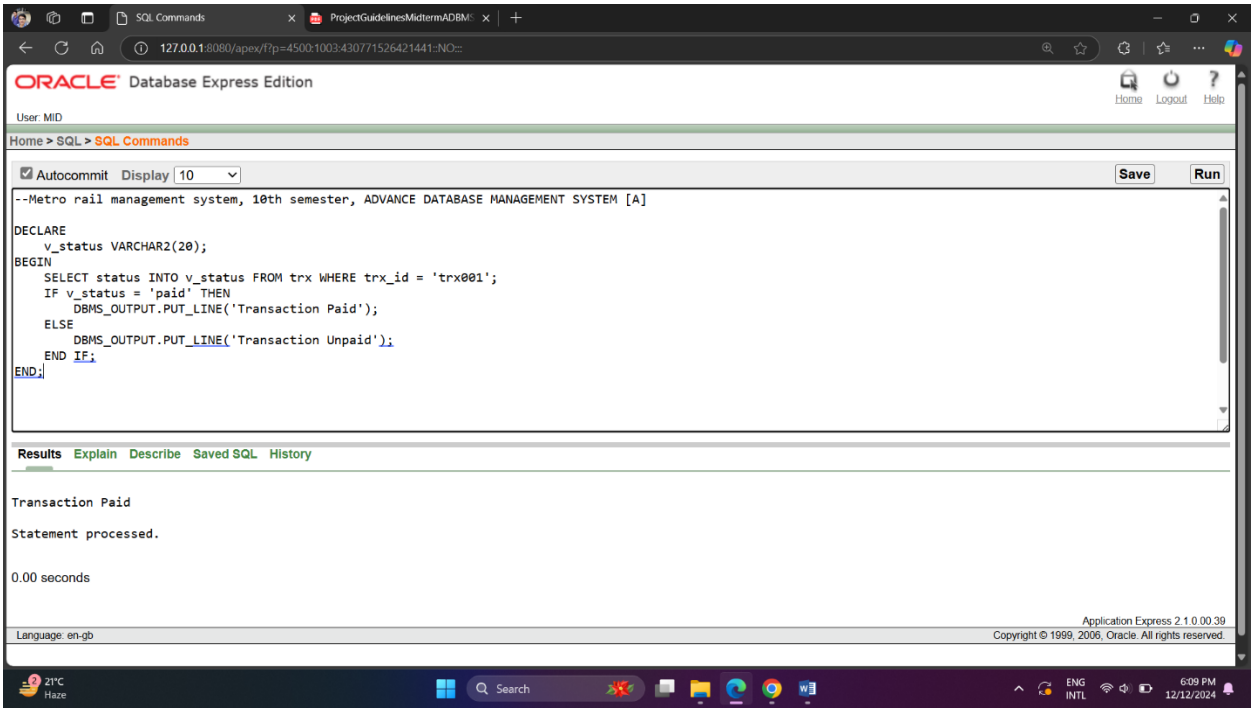
Conditional statements

a) Write a query that display transaction status from trx table

Ans:

```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

DECLARE
  v_status VARCHAR2(20);
BEGIN
  SELECT status INTO v_status FROM trx WHERE trx_id = 'trx001';
  IF v_status = 'paid' THEN
    DBMS_OUTPUT.PUT_LINE('Transaction Paid');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Transaction Unpaid');
  END IF;
END;
```



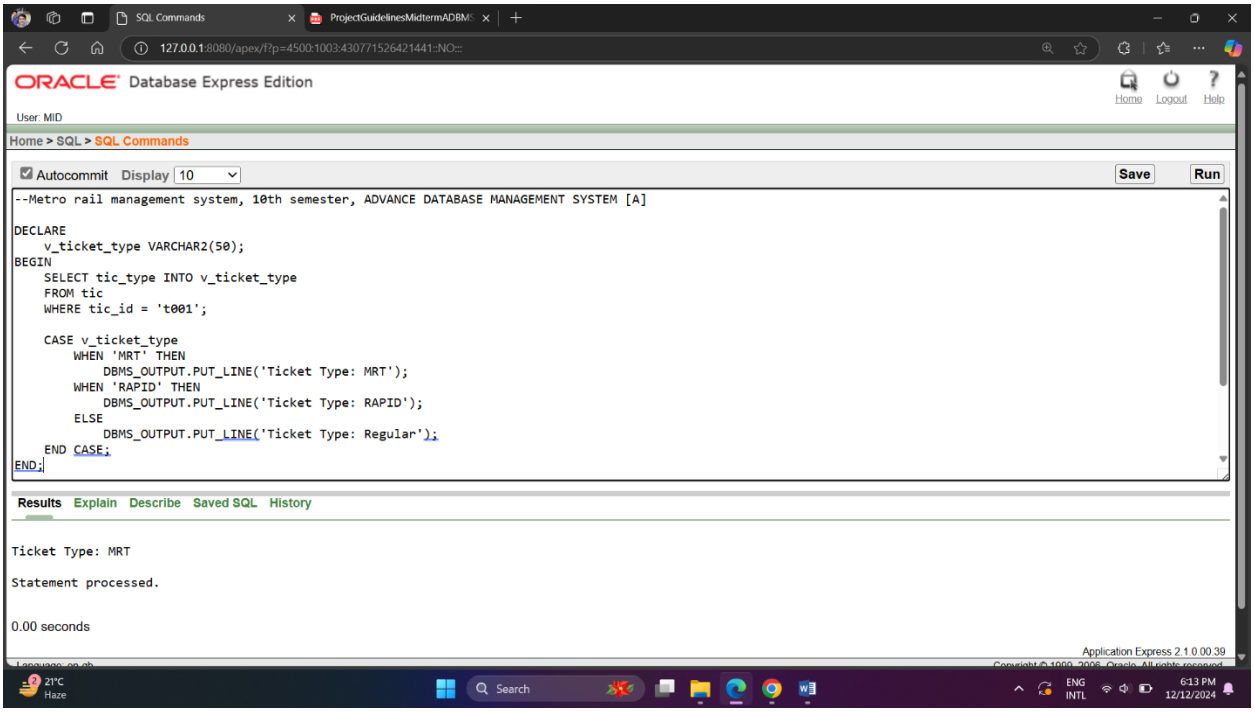
b) Write a query that display the ticket type using ticket id from tic table

Ans:

--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

```
DECLARE
  v_ticket_type VARCHAR2(50);
BEGIN
  SELECT tic_type INTO v_ticket_type
  FROM tic
  WHERE tic_id = 't001';

  CASE v_ticket_type
    WHEN 'MRT' THEN
      DBMS_OUTPUT.PUT_LINE('Ticket Type: MRT');
    WHEN 'RAPID' THEN
      DBMS_OUTPUT.PUT_LINE('Ticket Type: RAPID');
    ELSE
      DBMS_OUTPUT.PUT_LINE('Ticket Type: Regular');
  END CASE;
END;
```



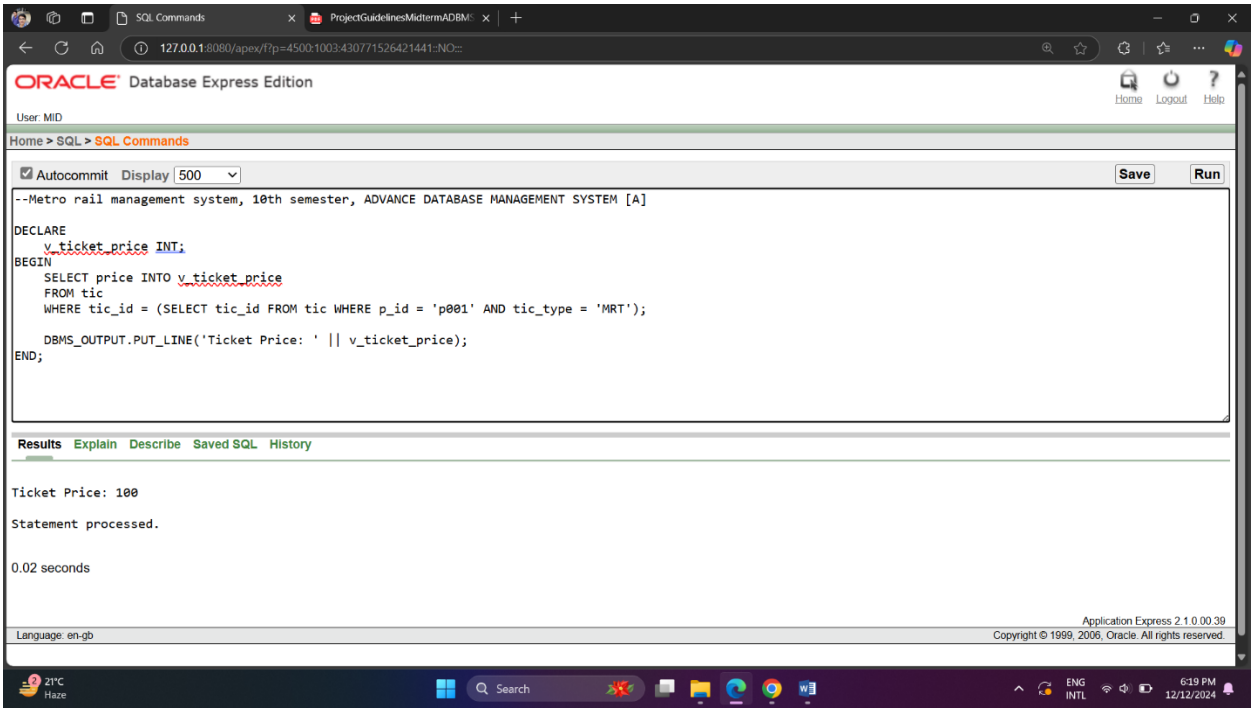
Subquery

a) Write a query that display ticket price from tic where passenger id p001 and ticket type MRT
Ans:

```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

DECLARE
  v_ticket_price INT;
BEGIN
  SELECT price INTO v_ticket_price
  FROM tic
  WHERE tic_id = (SELECT tic_id FROM tic WHERE p_id = 'p001' AND tic_type = 'MRT');

  DBMS_OUTPUT.PUT_LINE('Ticket Price: ' || v_ticket_price);
END;
```

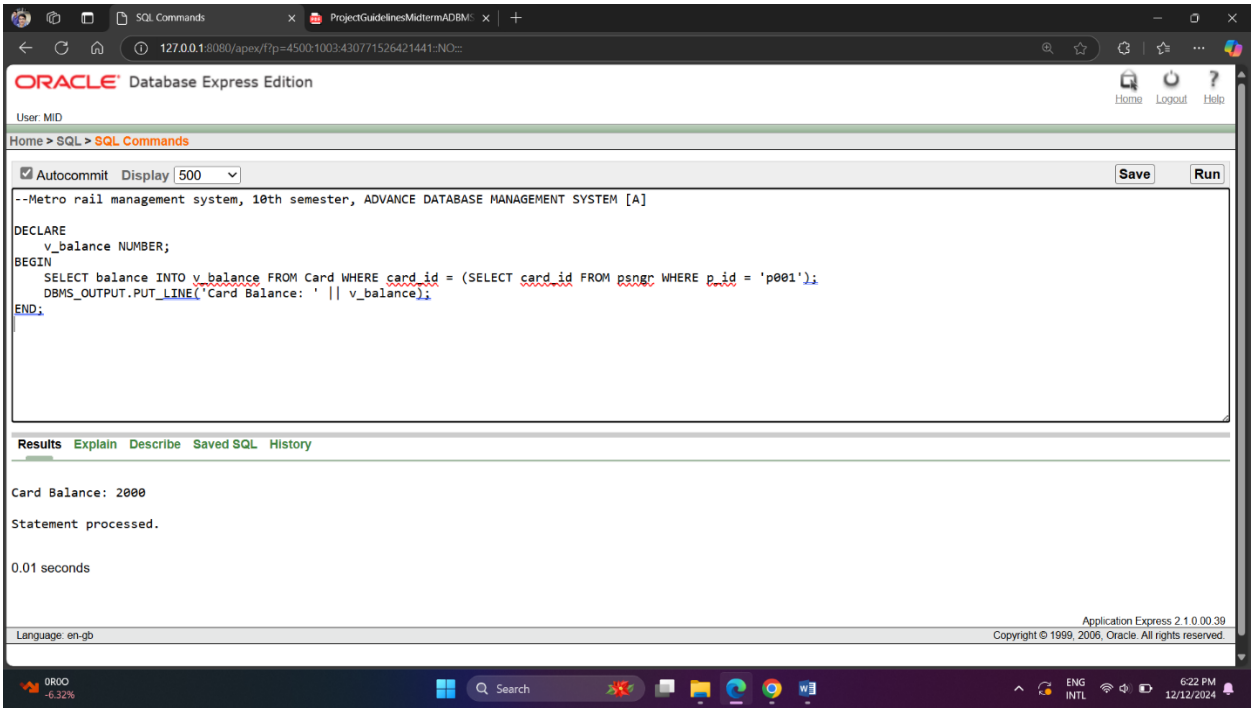


b) Write a query that display card balance from card table where p_id = p001

Ans:

```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

DECLARE
  v_balance NUMBER;
BEGIN
  SELECT balance INTO v_balance FROM Card WHERE card_id = (SELECT card_id FROM psngr WHERE p_id = 'p001');
  DBMS_OUTPUT.PUT_LINE('Card Balance: ' || v_balance);
END;
```



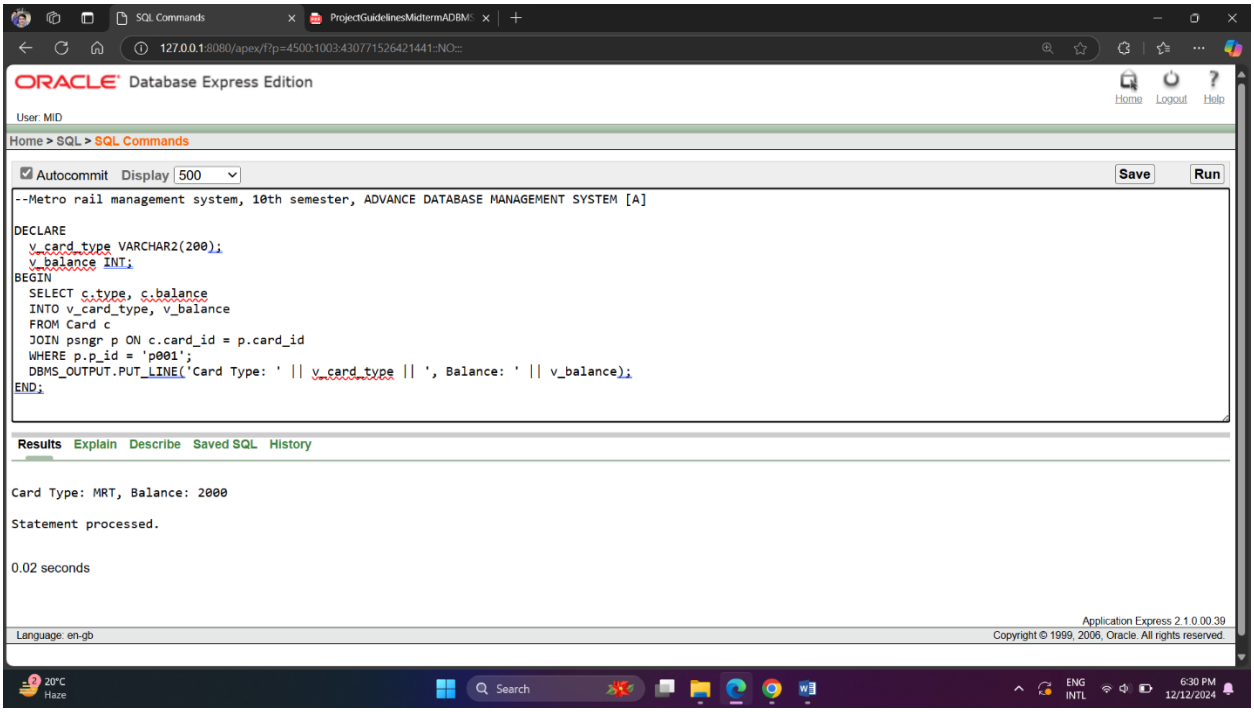
Joining

a) Write a query that display card type and balance from card and psngr using joining.

Ans:

--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

```
DECLARE
  v_card_type VARCHAR2(200);
  v_balance INT;
BEGIN
  SELECT c.type, c.balance
  INTO v_card_type, v_balance
  FROM Card c
  JOIN psngr p ON c.card_id = p.card_id
  WHERE p.p_id = 'p001';
  DBMS_OUTPUT.PUT_LINE('Card Type: ' || v_card_type || ', Balance: ' || v_balance);
END;
```

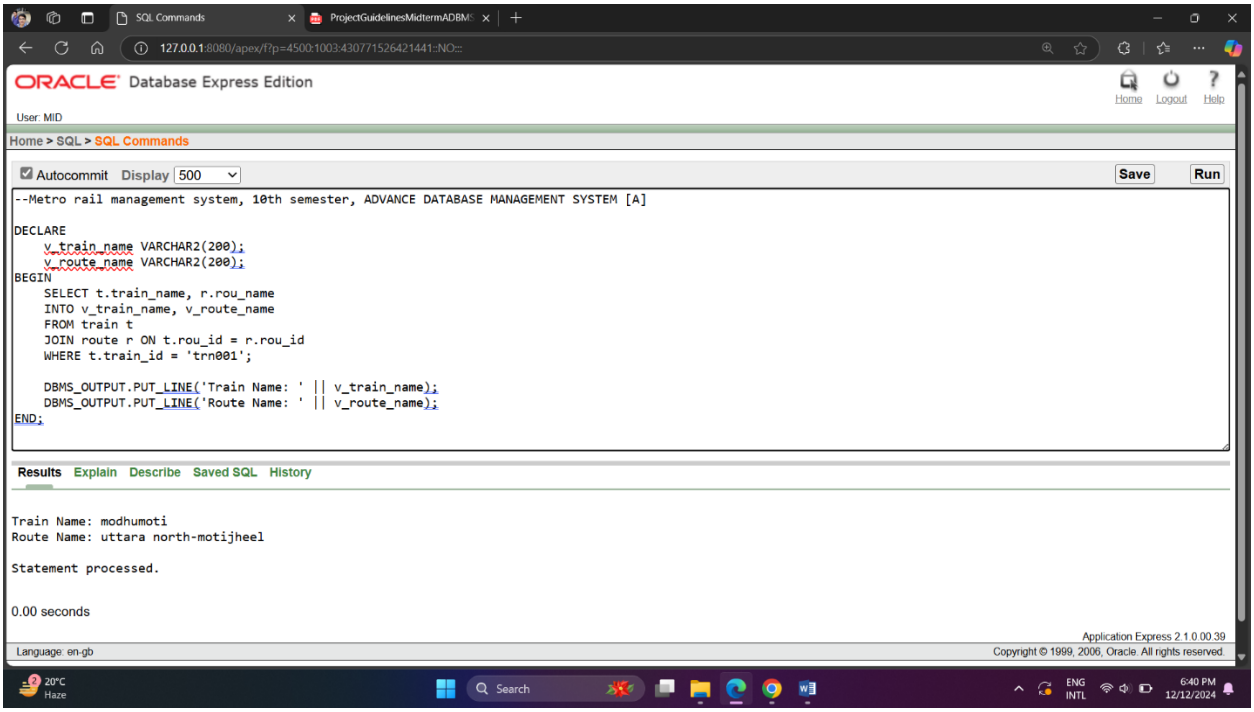


b) Write a query that display train name and route name from train and route table using joining
Ans:

```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

DECLARE
   v_train_name VARCHAR2(200);
   v_route_name VARCHAR2(200);
BEGIN
   SELECT t.train_name, r.rou_name
   INTO v_train_name, v_route_name
   FROM train t
   JOIN route r ON t.rou_id = r.rou_id
   WHERE t.train_id = 'trn001';

   DBMS_OUTPUT.PUT_LINE('Train Name: ' || v_train_name);
   DBMS_OUTPUT.PUT_LINE('Route Name: ' || v_route_name);
END;
```



Advance PL/SQL

stored function

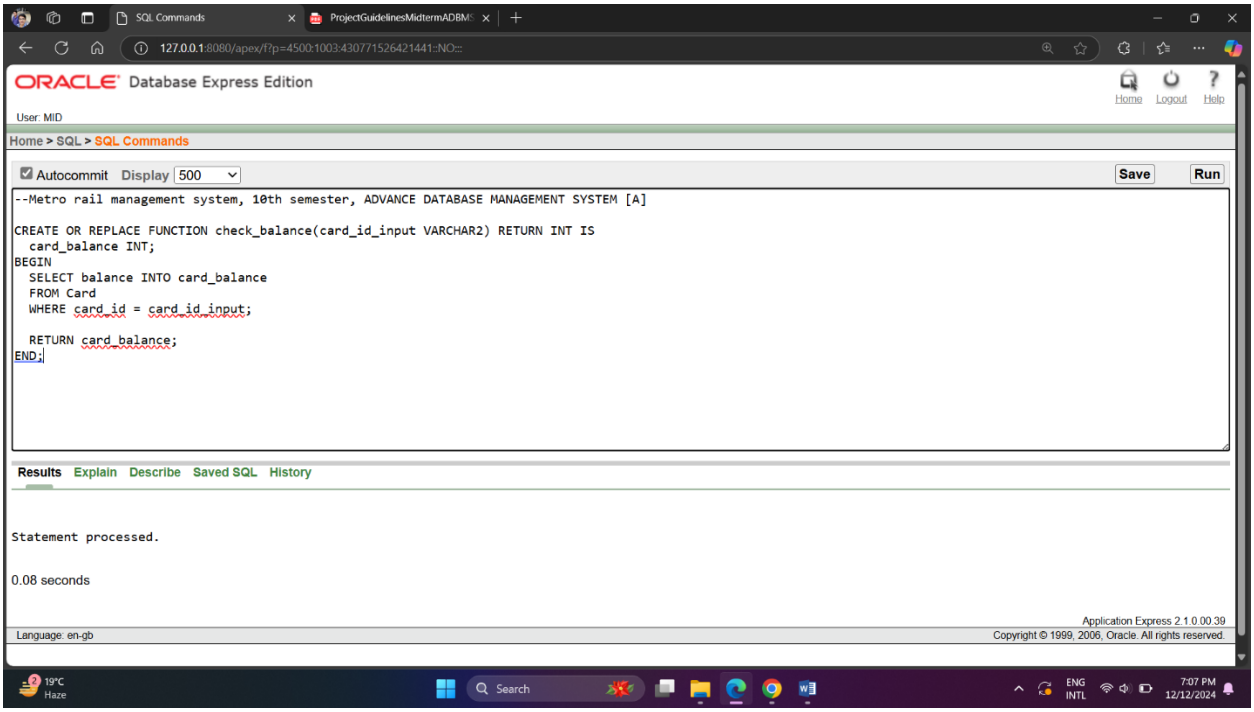
a) Write a query to create a function to check card balance

Ans:

--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

```
CREATE OR REPLACE FUNCTION check_balance(card_id_input VARCHAR2) RETURN INT IS
  card_balance INT;
BEGIN
  SELECT balance INTO card_balance
  FROM Card
  WHERE card_id = card_id_input;

  RETURN card_balance;
END;
```



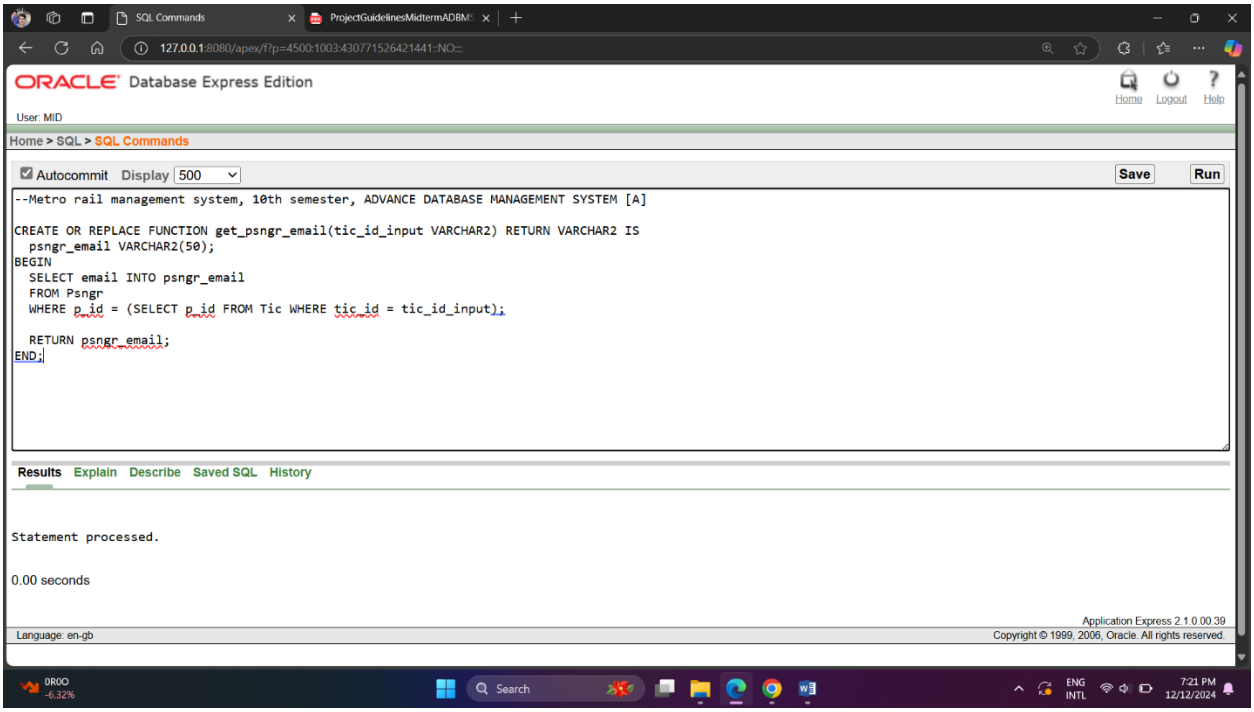
b) Write a query to create a Function to get passenger email by ticket ID

Ans:

```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

CREATE OR REPLACE FUNCTION get_psngr_email(tic_id_input VARCHAR2) RETURN VARCHAR2 IS
  psngr_email VARCHAR2(50);
BEGIN
  SELECT email INTO psngr_email
  FROM Psngr
  WHERE p_id = (SELECT p_id FROM Tic WHERE tic_id = tic_id_input);

  RETURN psngr_email;
END;
```



Stored procedure

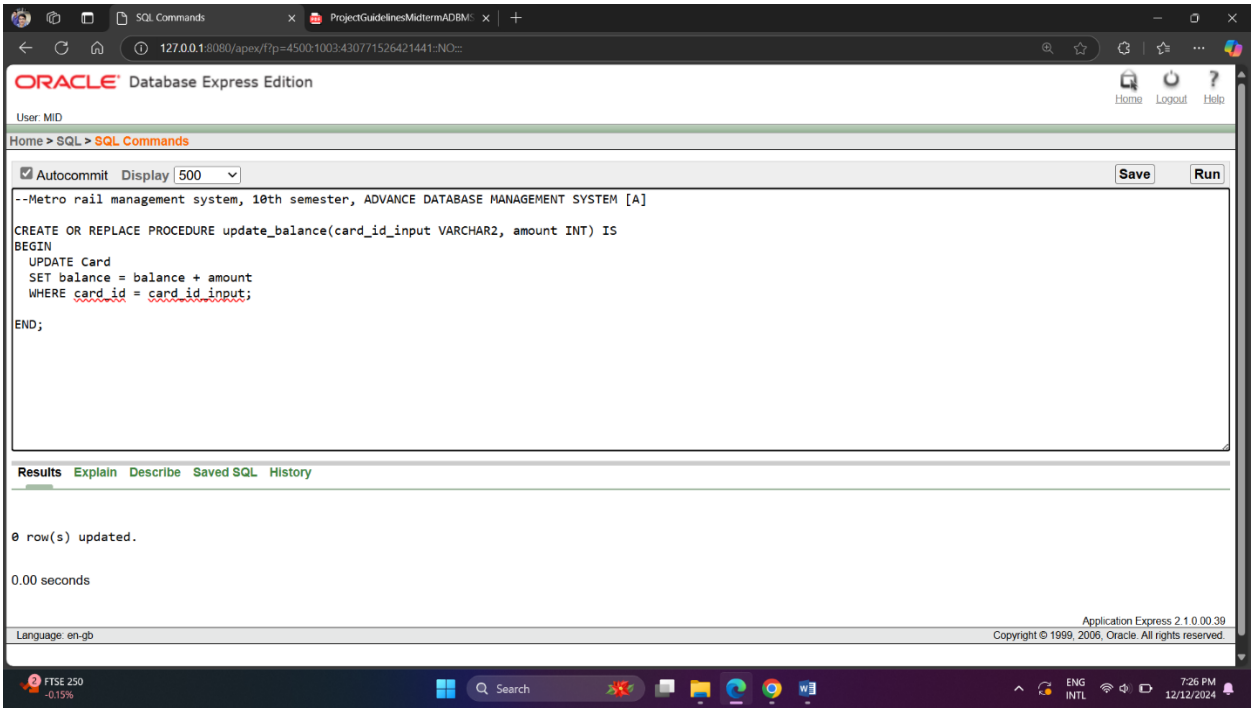
a) Write a query to create Procedure to update card balance

Ans:

--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

```
CREATE OR REPLACE PROCEDURE update_balance(card_id_input VARCHAR2, amount INT) IS
BEGIN
  UPDATE Card
  SET balance = balance + amount
  WHERE card_id = card_id_input;

END;
```



b) Write a query to create Procedure to mark a transaction as paid

Ans:

```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

CREATE OR REPLACE PROCEDURE mark_transaction_paid(trx_id_input VARCHAR2) IS
BEGIN
  UPDATE Trx
  SET status = 'paid'
  WHERE trx_id = trx_id_input;
END;
```

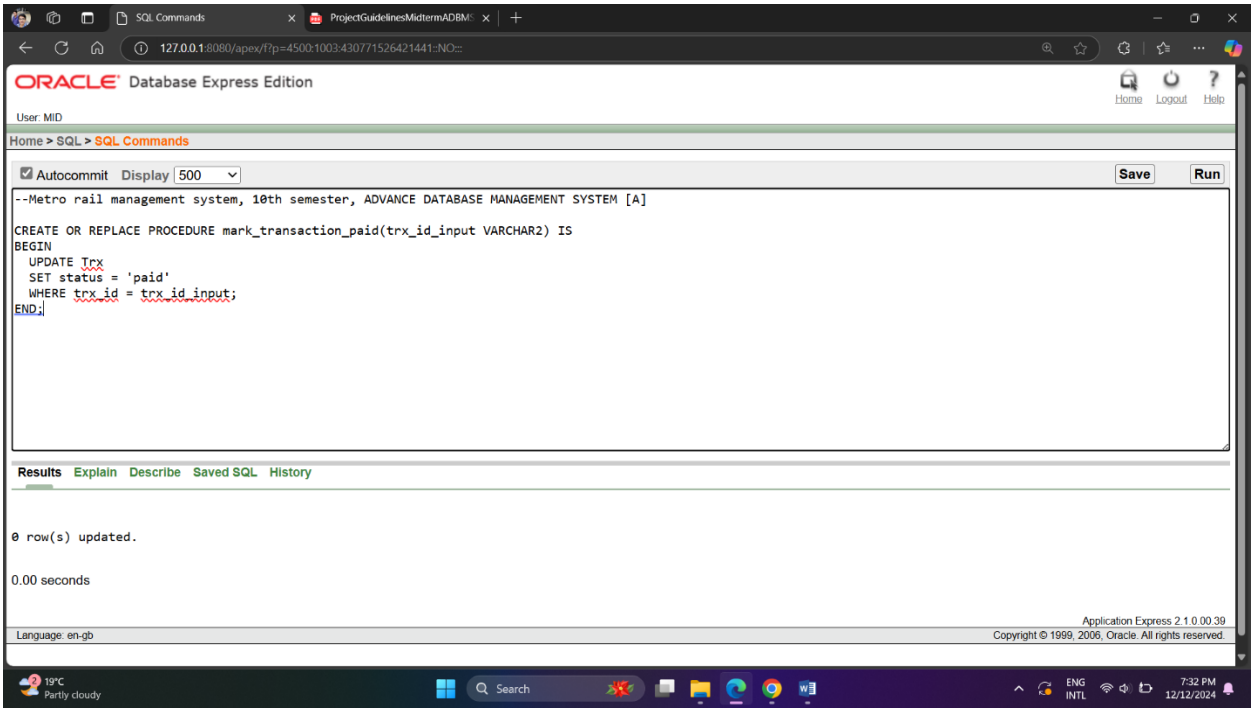


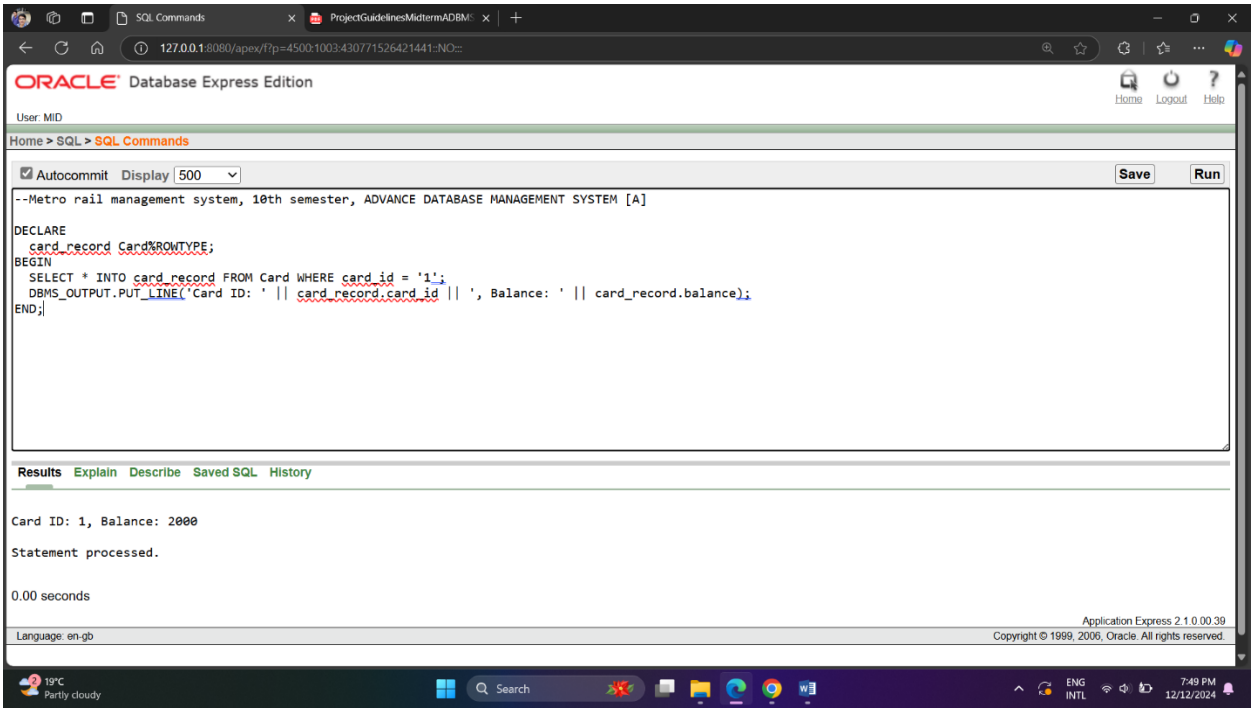
Table-based record

a) Write a query to find record for Card table

Ans:

--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

```
DECLARE
 card_record Card%ROWTYPE;
BEGIN
 SELECT * INTO card_record FROM Card WHERE card_id = '1';
 DBMS_OUTPUT.PUT_LINE('Card ID: ' || card_record.card_id || ', Balance: ' ||
card_record.balance);
END;
```

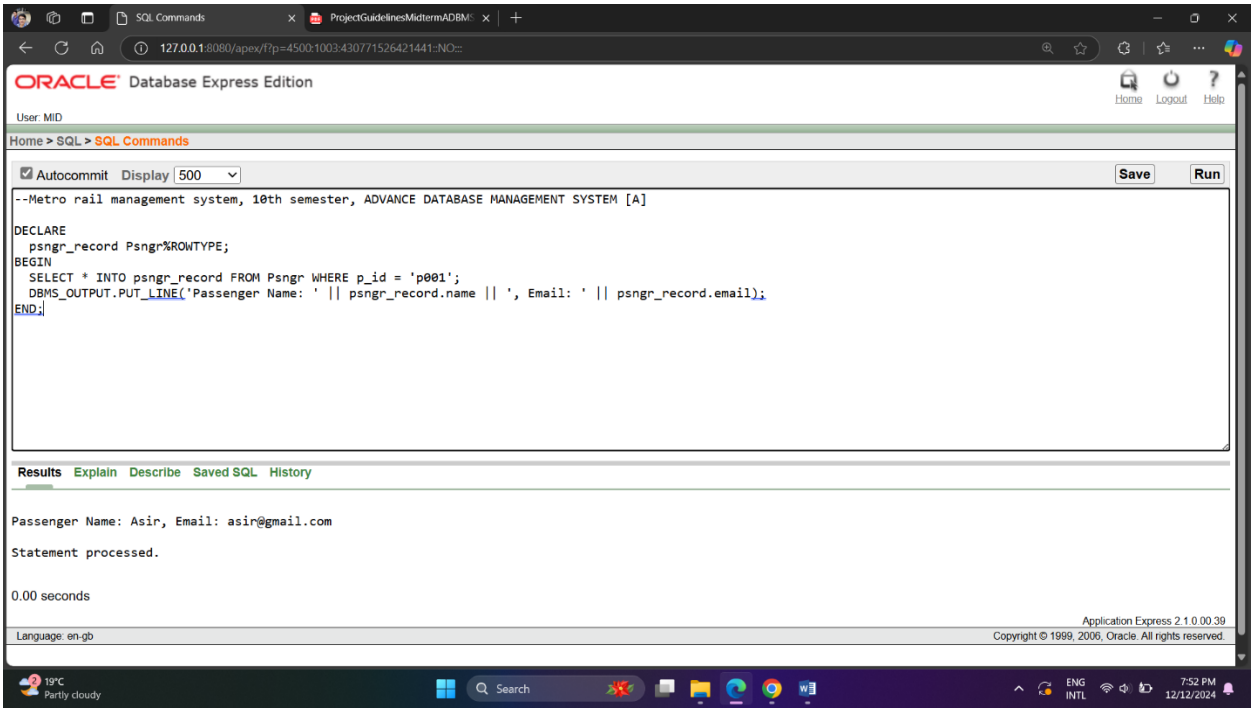


b) Write a query to find record for Passenger table

Ans:

--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

```
DECLARE
  psngr_record Psngr%ROWTYPE;
BEGIN
  SELECT * INTO psngr_record FROM Psngr WHERE p_id = 'p001';
  DBMS_OUTPUT.PUT_LINE('Passenger Name: ' || psngr_record.name || ', Email: ' ||
psngr_record.email);
END;
```



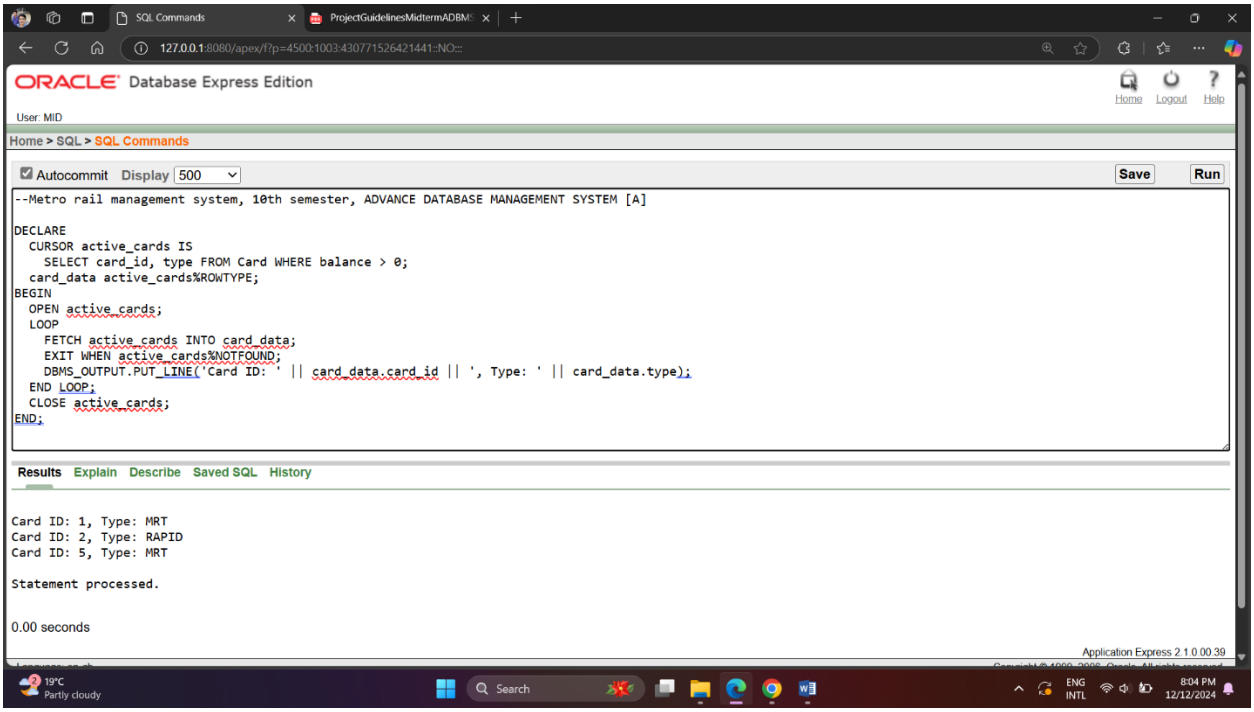
Explicit cursor

a) Create a explicit cursor that can output all active cards from card table

Ans:

```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM
[A]

DECLARE
  CURSOR active_cards IS
    SELECT card_id, type FROM Card WHERE balance > 0;
  card_data active_cards%ROWTYPE;
BEGIN
  OPEN active_cards;
  LOOP
    FETCH active_cards INTO card_data;
    EXIT WHEN active_cards%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Card ID: ' || card_data.card_id || ', Type: ' || card_data.type);
  END LOOP;
  CLOSE active_cards;
END;
```

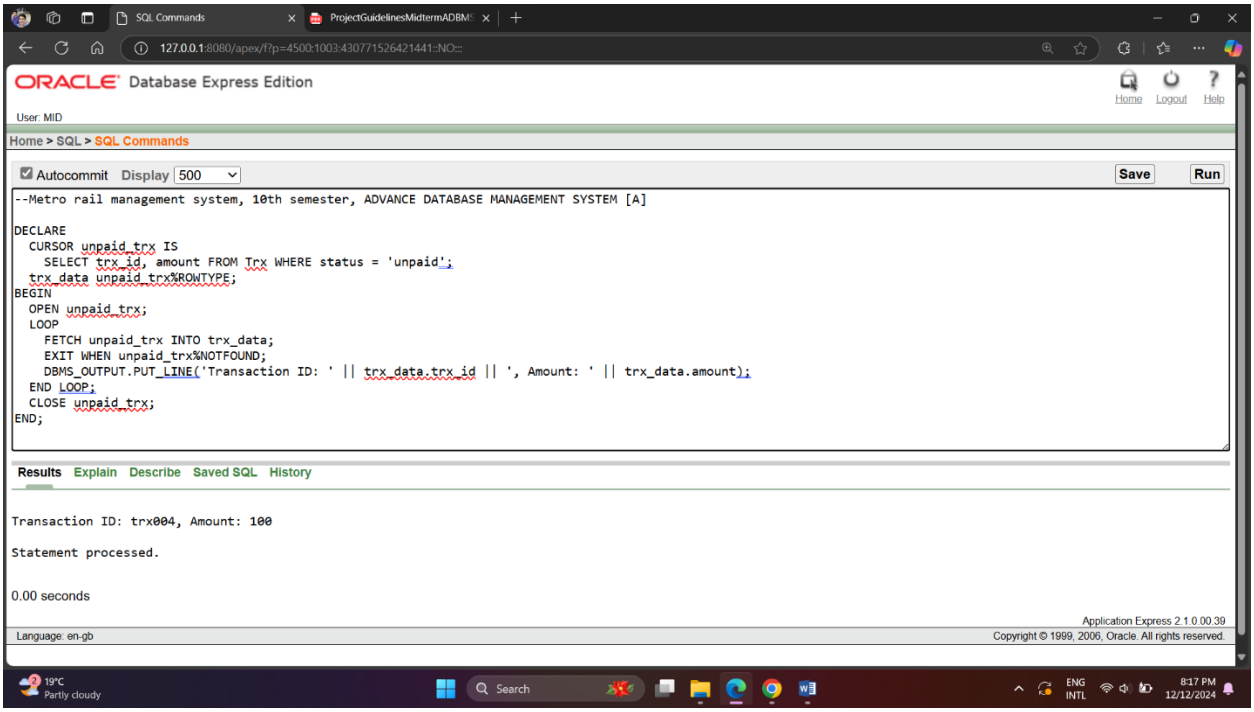


b) Create a explicit cursor that can output all Transaction ID and Amount with unpaid status from Trx table

Ans:

--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

```
DECLARE
  CURSOR unpaid_trx IS
    SELECT trx_id, amount FROM Trx WHERE status = 'unpaid';
  trx_data unpaid_trx%ROWTYPE;
BEGIN
  OPEN unpaid_trx;
  LOOP
    FETCH unpaid_trx INTO trx_data;
    EXIT WHEN unpaid_trx%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Transaction ID: ' || trx_data.trx_id || ', Amount: ' ||
trx_data.amount);
  END LOOP;
  CLOSE unpaid_trx;
END;
```



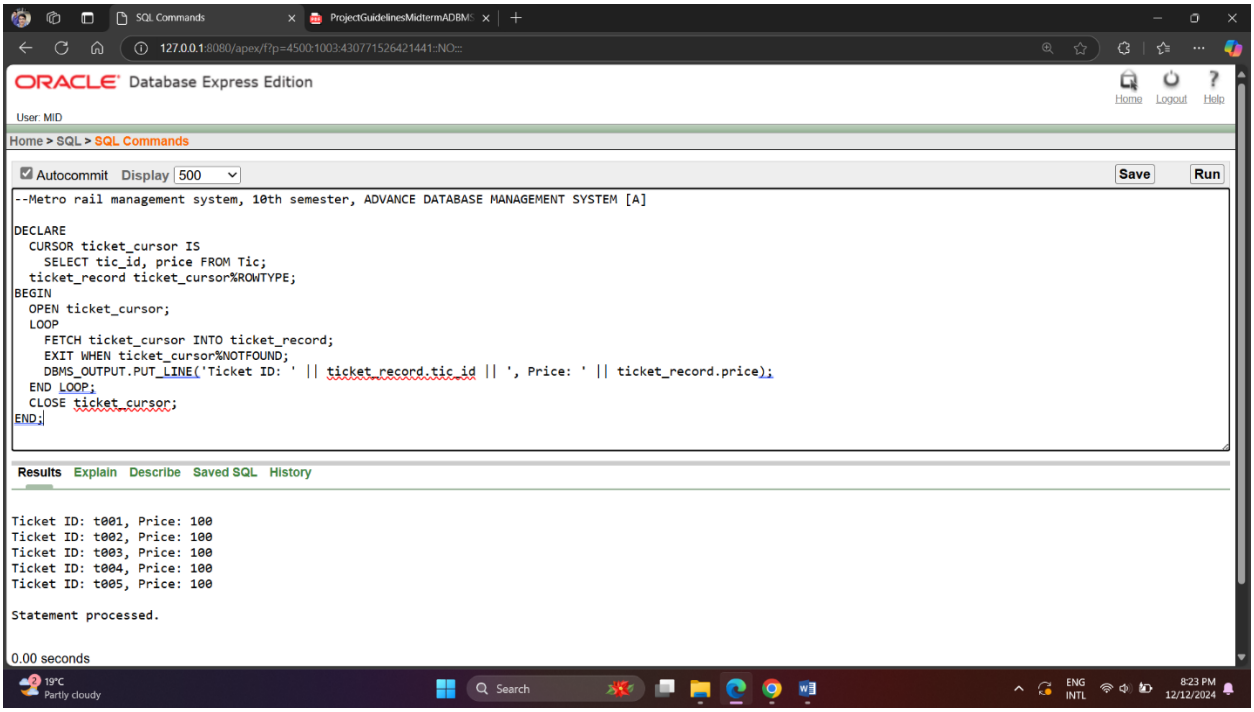
Cursor-based record

a) Create a record that can output ticked id and price from tic table

Ans:

--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]

```
DECLARE
  CURSOR ticket_cursor IS
    SELECT tic_id, price FROM Tic;
  ticket_record ticket_cursor%ROWTYPE;
BEGIN
  OPEN ticket_cursor;
  LOOP
    FETCH ticket_cursor INTO ticket_record;
    EXIT WHEN ticket_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Ticket ID: ' || ticket_record.tic_id || ', Price: ' ||
ticket_record.price);
  END LOOP;
  CLOSE ticket_cursor;
END;
```

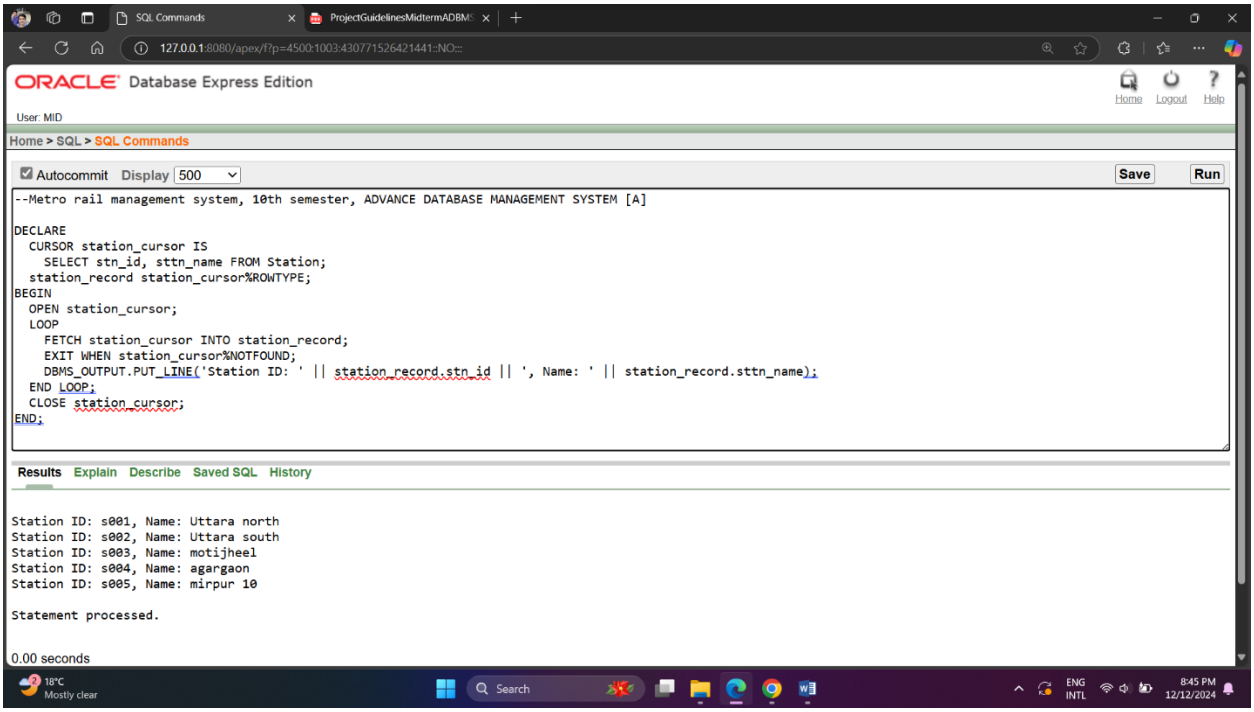


b) Create a record that can output Station id and name from station table

Ans:

```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM
[A]

DECLARE
 CURSOR station_cursor IS
  SELECT stn_id, sttn_name FROM Station;
 station_record station_cursor%ROWTYPE;
BEGIN
 OPEN station_cursor;
 LOOP
  FETCH station_cursor INTO station_record;
  EXIT WHEN station_cursor%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE('Station ID: ' || station_record.stn_id || ', Name: ' ||
station_record.sttn_name);
 END LOOP;
 CLOSE station_cursor;
END;
```



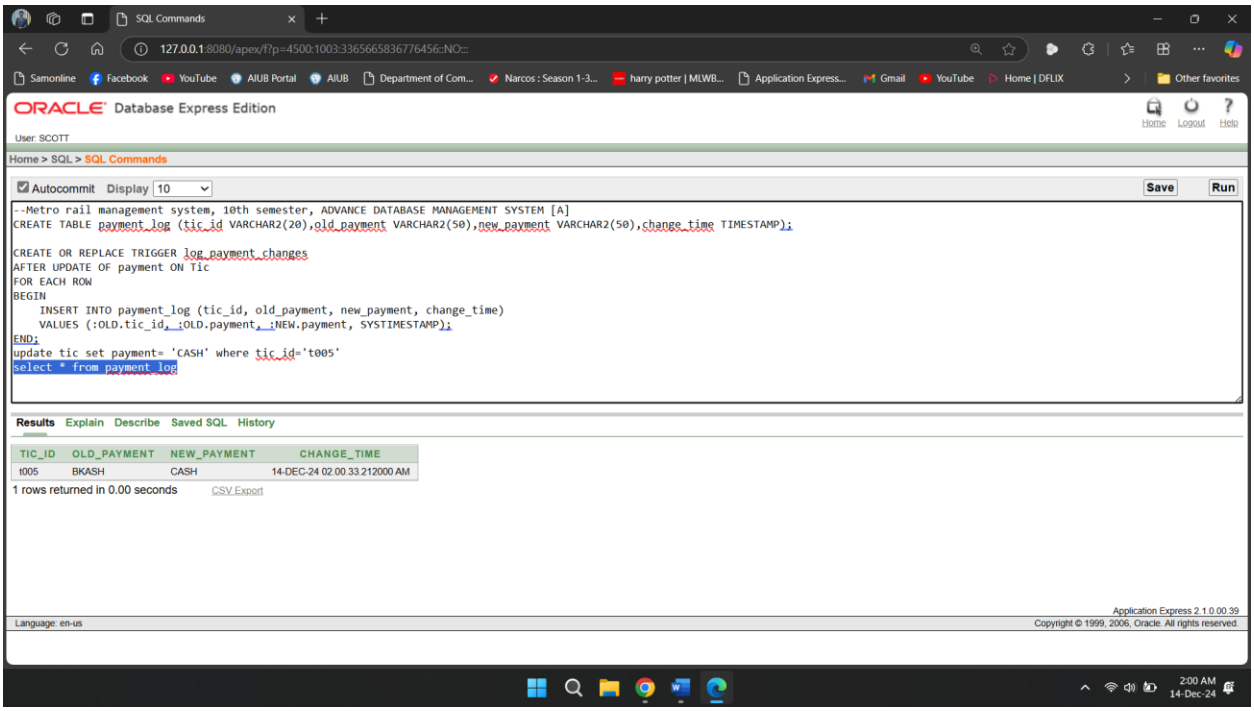
Row level trigger

a) Create a trigger to keep a payment log whenever payment method is changed.

Ans:

```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]
CREATE TABLE payment_log (tic_id VARCHAR2(20),old_payment VARCHAR2(50),new_payment VARCHAR2(50),change_time TIMESTAMP);

CREATE OR REPLACE TRIGGER log_payment_changes
AFTER UPDATE OF payment ON Tic
FOR EACH ROW
BEGIN
   INSERT INTO payment_log (tic_id, old_payment, new_payment, change_time)
   VALUES (:OLD.tic_id, :OLD.payment, :NEW.payment, SYSTIMESTAMP);
END;
update tic set payment= 'CASH' where tic_id='t005'
select * from payment_log
```

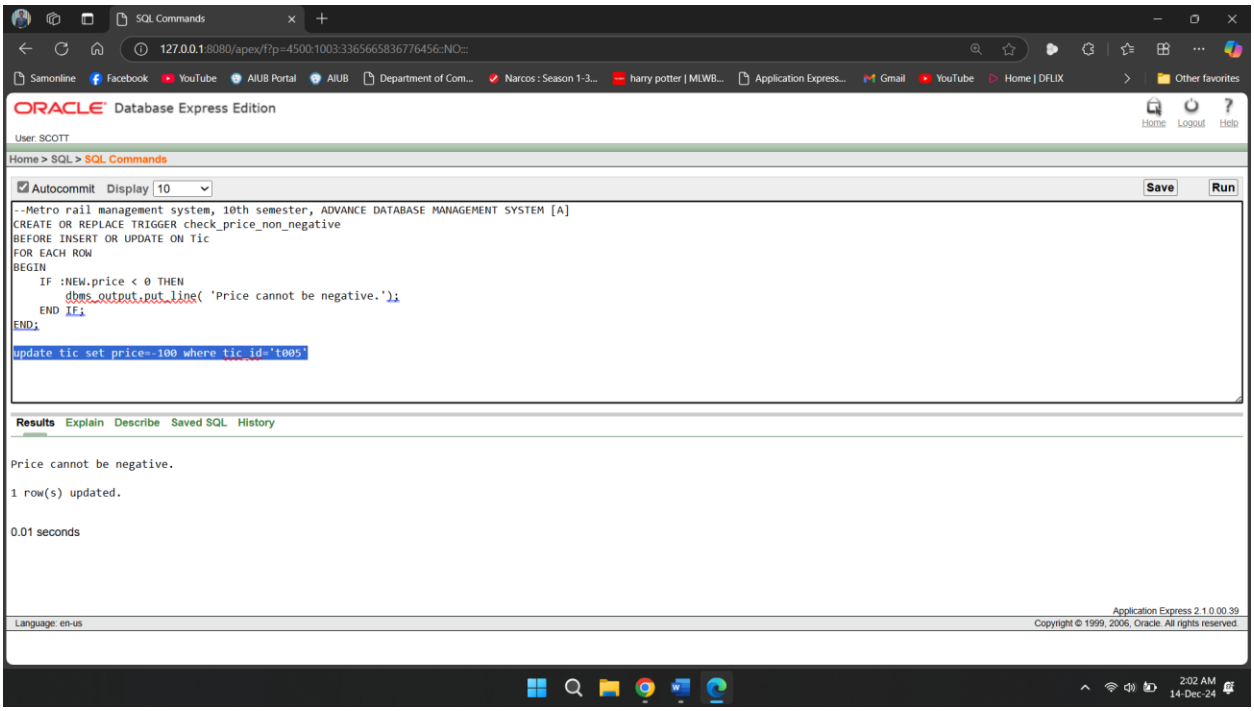


b) Create a trigger to stop the price from being negative.

Ans:

```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]
CREATE OR REPLACE TRIGGER check_price_non_negative
BEFORE INSERT OR UPDATE ON Tic
FOR EACH ROW
BEGIN
  IF :NEW.price < 0 THEN
    dbms_output.put_line( 'Price cannot be negative.');
  END IF;
END;

update tic set price=-100 where tic_id='t005'
```



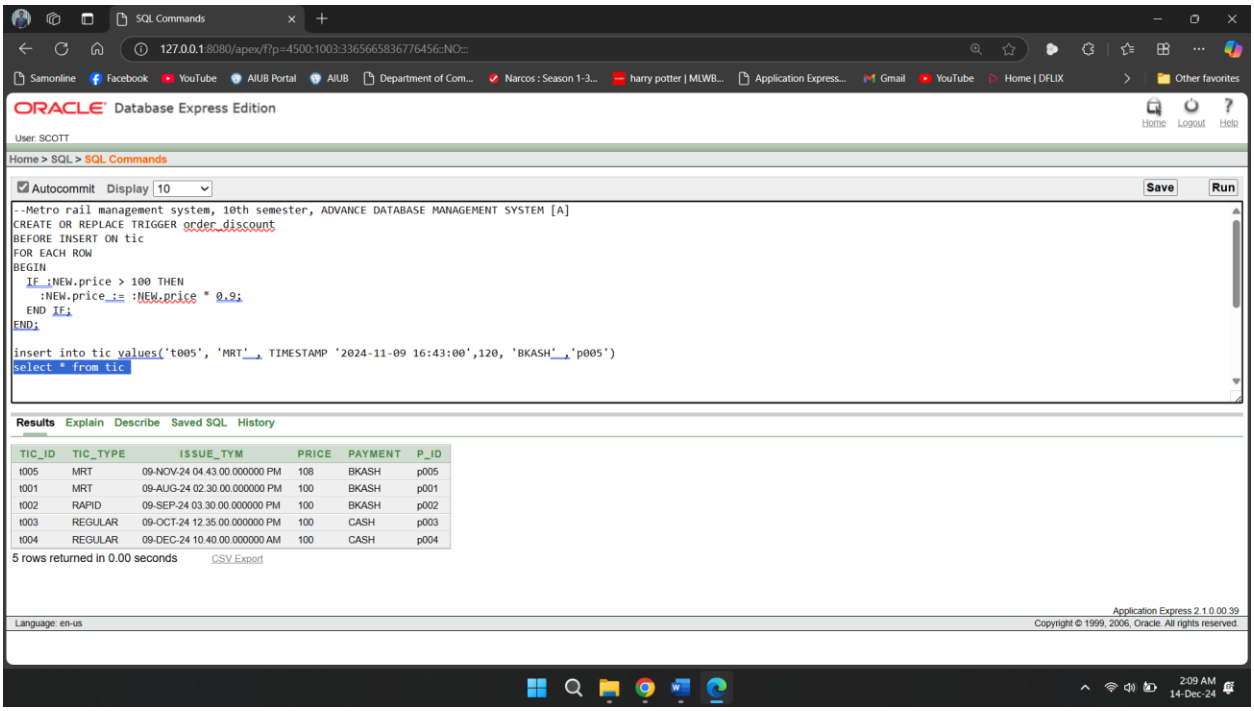
statement level trigger

a) Create a trigger to give a discount everytime the price is more than 100.

Ans:

```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM
[A]
CREATE OR REPLACE TRIGGER order_discount
BEFORE INSERT ON tic
FOR EACH ROW
BEGIN
  IF :NEW.price > 100 THEN
    :NEW.price := :NEW.price * 0.9;
  END IF;
END;

insert into tic values('t005', 'MRT' , TIMESTAMP '2024-11-09 16:43:00',120, 'BKASH' ,'p005')
select * from tic
```

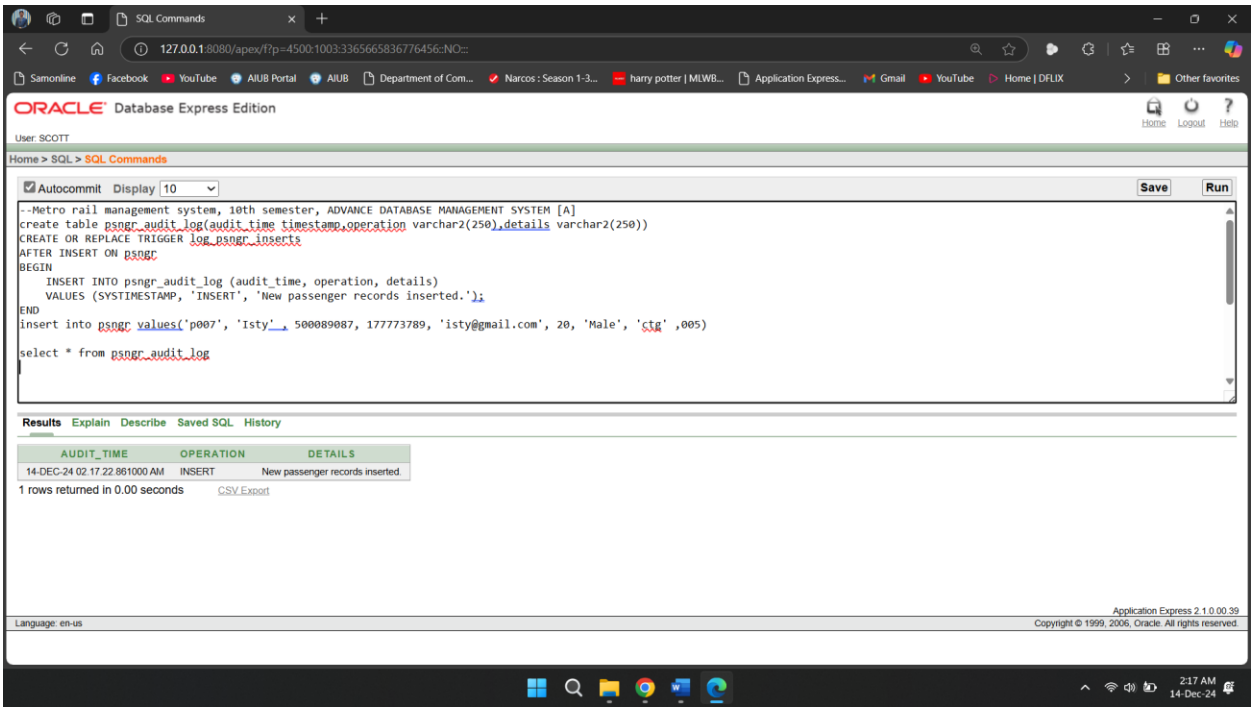


b) Create a trigger

Ans:

```
--Metro rail management system, 10th semester, ADVANCE DATABASE MANAGEMENT SYSTEM [A]
create table psngr_audit_log(audit_time timestamp,operation varchar2(250),details varchar2(250))
CREATE OR REPLACE TRIGGER log_psngr_inserts
AFTER INSERT ON psngr
BEGIN
   INSERT INTO psngr_audit_log (audit_time, operation, details)
   VALUES (SYSTIMESTAMP, 'INSERT', 'New passenger records inserted.');
END
insert into psngr values('p007', 'Isty' , 500089087, 177773789, 'isty@gmail.com', 20, 'Male', 'ctg' ,005)

select * from psngr_audit_log
```



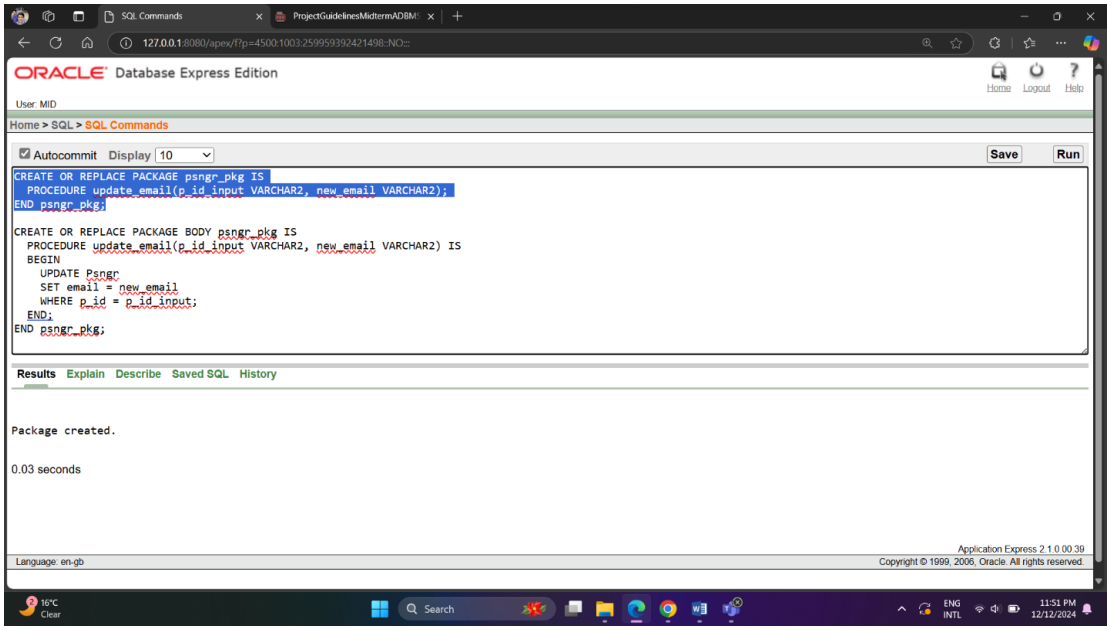
Package

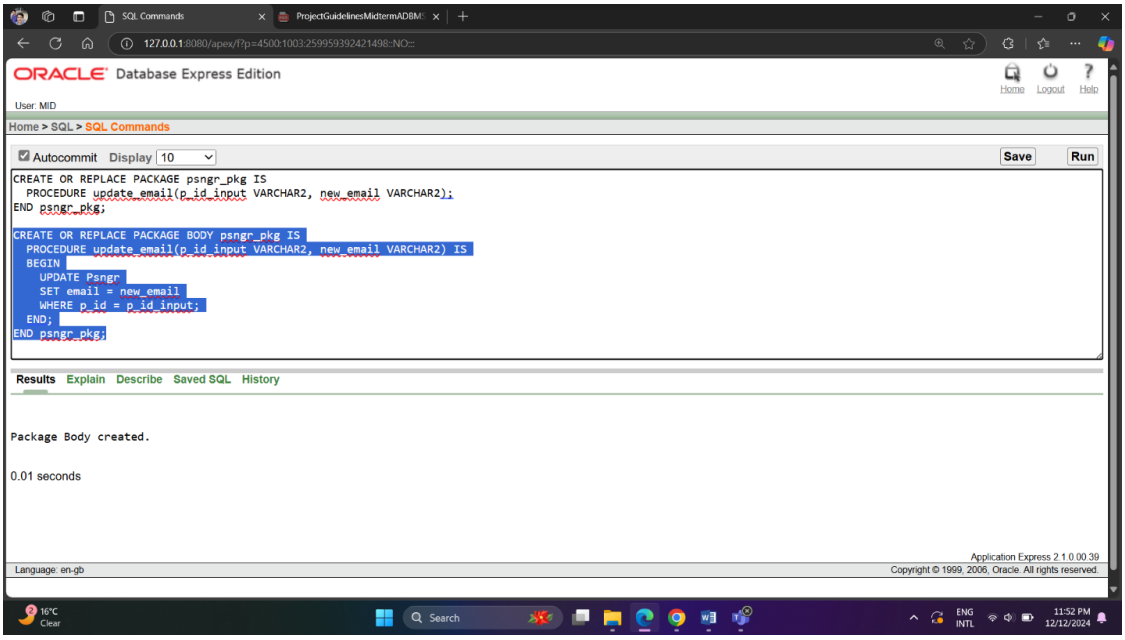
a) Create a package that contains a procedure which can update passenger email id.

Ans:

```
CREATE OR REPLACE PACKAGE psngr_pkg IS
  PROCEDURE update_email(p_id_input VARCHAR2, new_email VARCHAR2);
END psngr_pkg;

CREATE OR REPLACE PACKAGE BODY psngr_pkg IS
  PROCEDURE update_email(p_id_input VARCHAR2, new_email VARCHAR2) IS
  BEGIN
    UPDATE Psngr
    SET email = new_email
    WHERE p_id = p_id_input;
  END;
END psngr_pkg;
```





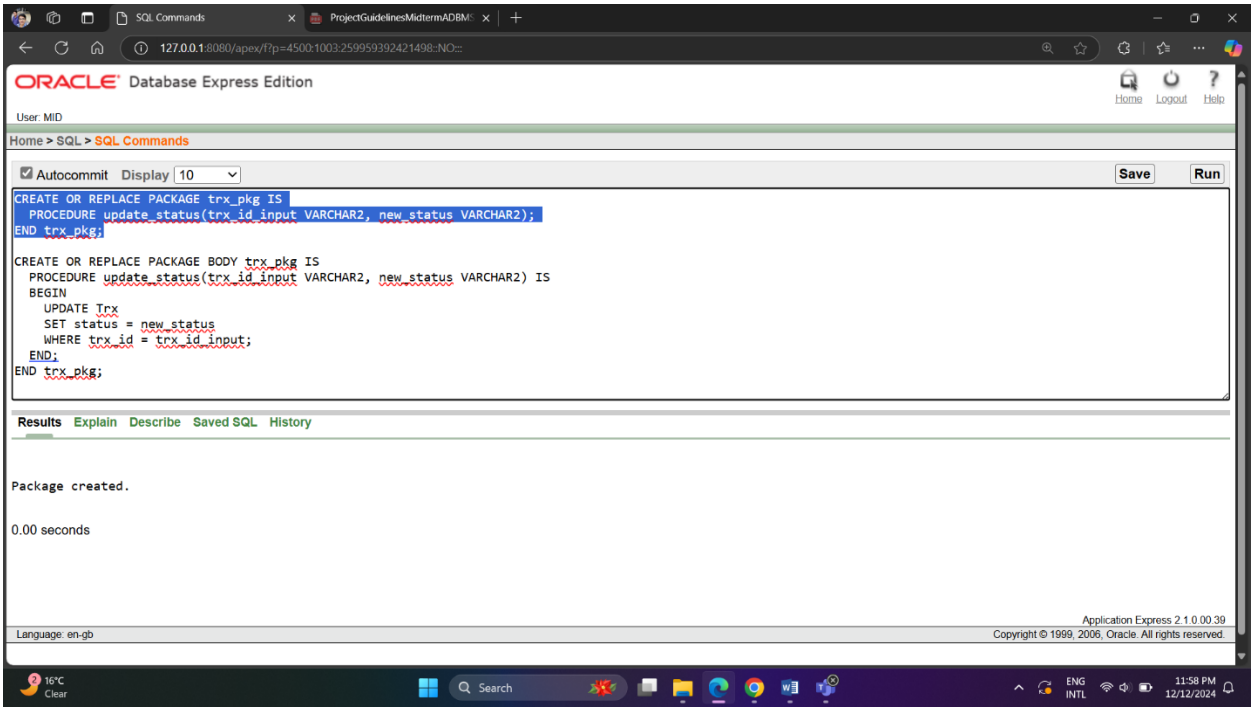
b) Create a package that contains a procedure which can update trx status.

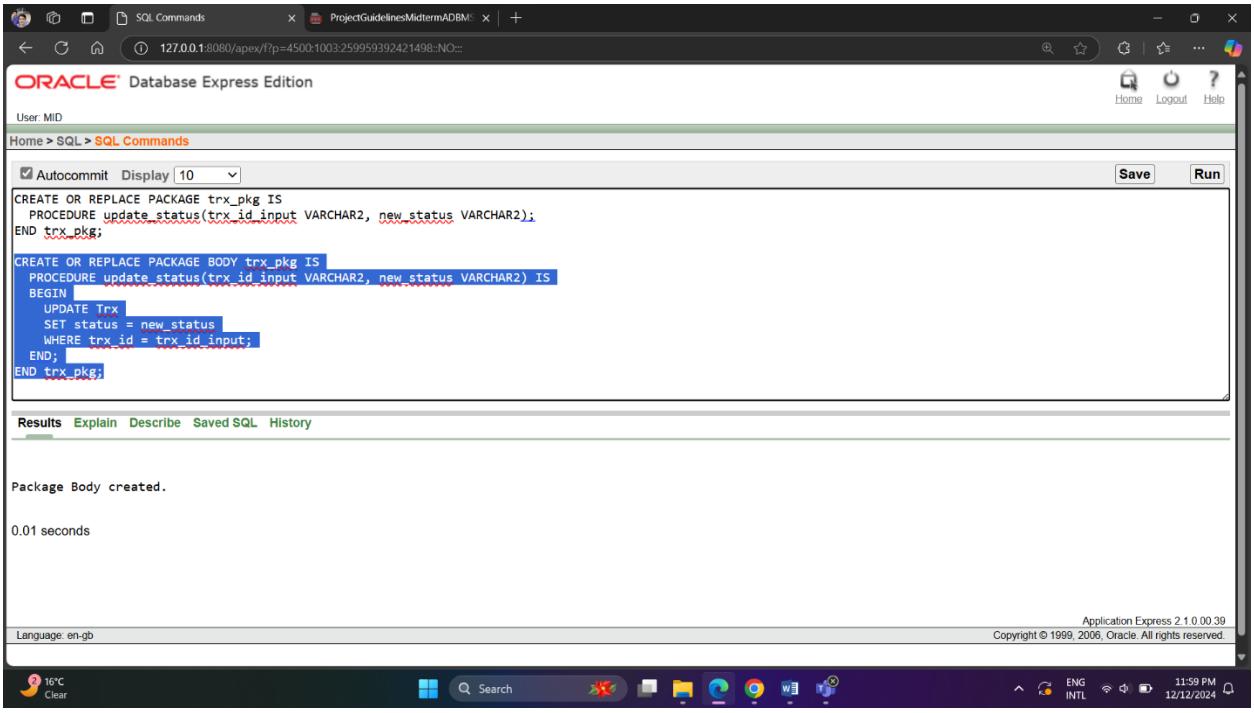
Ans:

```
CREATE OR REPLACE PACKAGE trx_pkg IS
  PROCEDURE update_status(trx_id_input VARCHAR2, new_status VARCHAR2);
END trx_pkg;

CREATE OR REPLACE PACKAGE BODY trx_pkg IS
  PROCEDURE update_status(trx_id_input VARCHAR2, new_status VARCHAR2) IS
```

```
 BEGIN
   UPDATE Trx
   SET status = new_status
   WHERE trx_id = trx_id_input;
 END;
END trx_pkg;
```



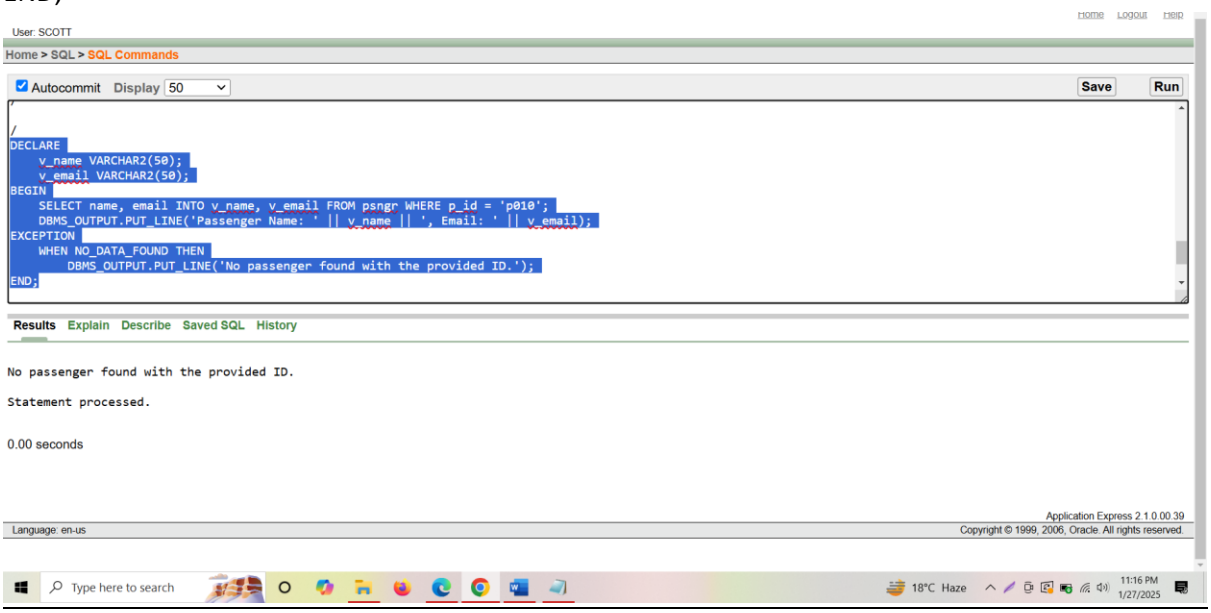


Exception handling

a) How can you handle a situation where a query retrieves no rows (e.g., no passenger is found with a specific (p_id) and display an appropriate message?

Ans:

```
DECLARE
  v_name VARCHAR2(50);
  v_email VARCHAR2(50);
BEGIN
  SELECT name, email INTO v_name, v_email FROM psngr WHERE p_id = 'p010';
  DBMS_OUTPUT.PUT_LINE('Passenger Name: ' || v_name || ', Email: ' || v_email);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No passenger found with the provided ID.');
END;
```

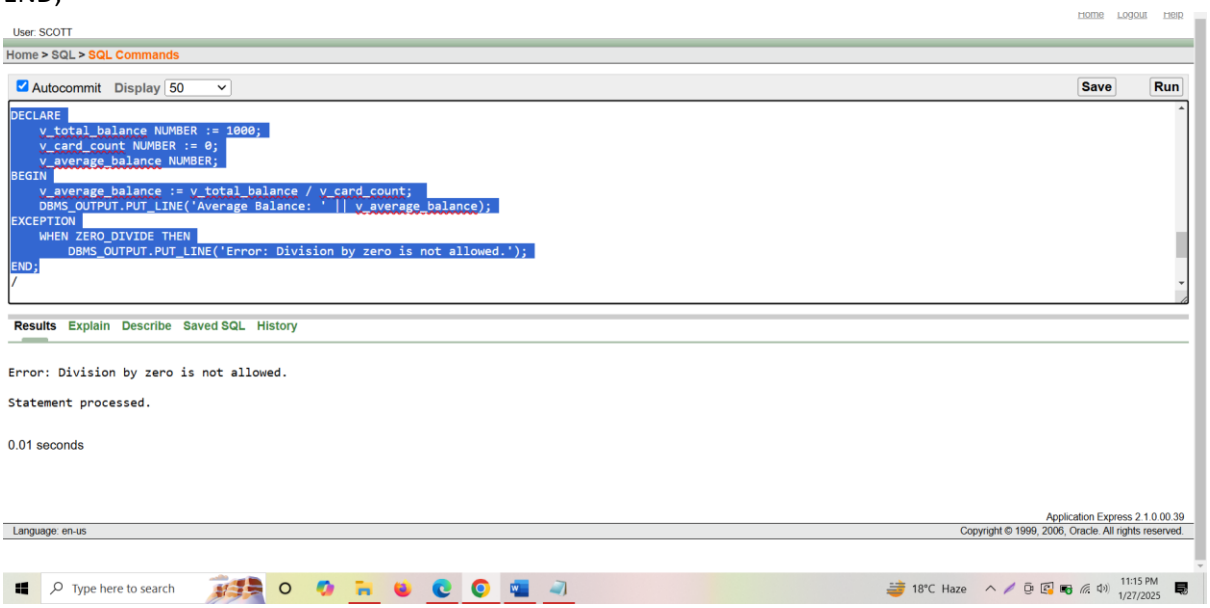


b) How can you handle a divide-by-zero error when calculating an average balance in a scenario where the total number of cards might be zero?

Ans:

```
DECLARE
  v_total_balance NUMBER := 1000;
  v_card_count NUMBER := 0;
  v_average_balance NUMBER;
BEGIN
  v_average_balance := v_total_balance / v_card_count;
  DBMS_OUTPUT.PUT_LINE('Average Balance: ' || v_average_balance);
EXCEPTION
```

```
    WHEN ZERO_DIVIDE THEN
        DBMS_OUTPUT.PUT_LINE('Error: Division by zero is not allowed.');
END;
```

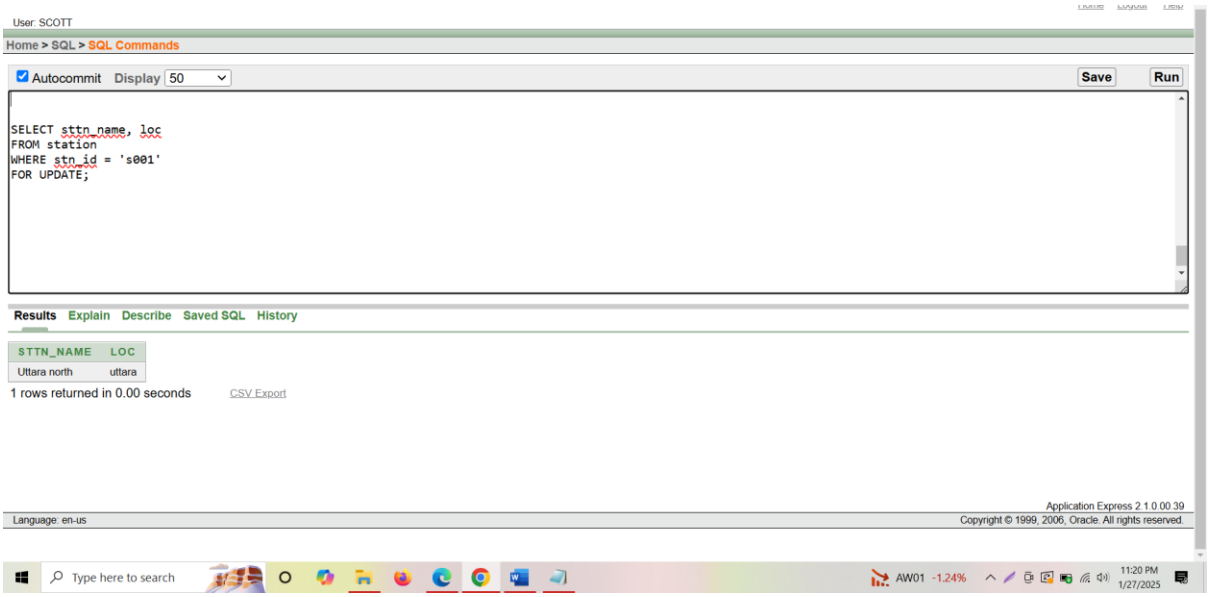


Implicit Locking

a) How can you lock specific rows in the station table to prevent other users from updating or deleting them while ensuring you can update them?

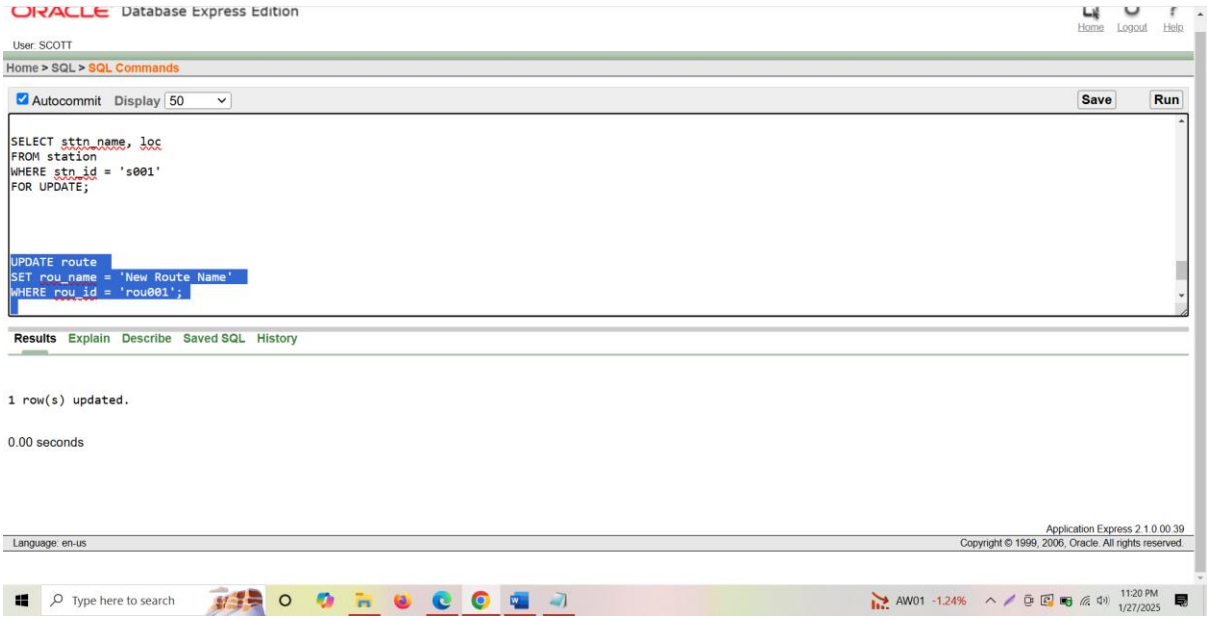
Ans:

```
SELECT sttn_name, loc
FROM station
WHERE stn_id = 's001'
FOR UPDATE;
```



b) How can you ensure no other user modifies a specific route while updating its name in the route table?

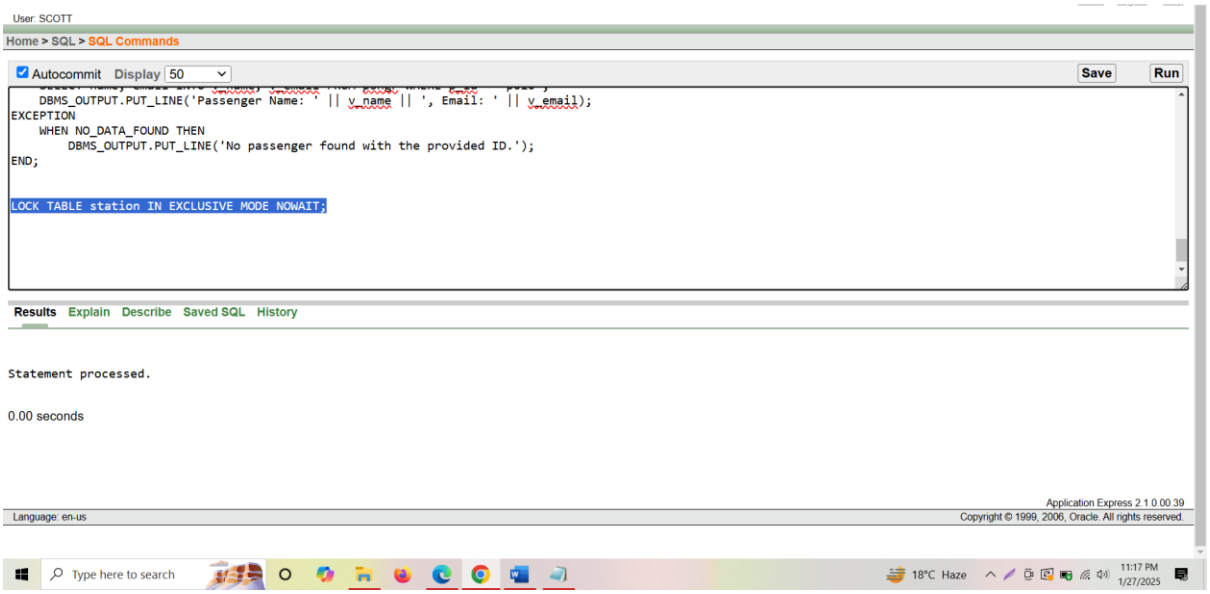
Ans:
UPDATE route
SET rou_name = 'New Route Name'
WHERE rou_id = 'rou001';



Explicit Locking

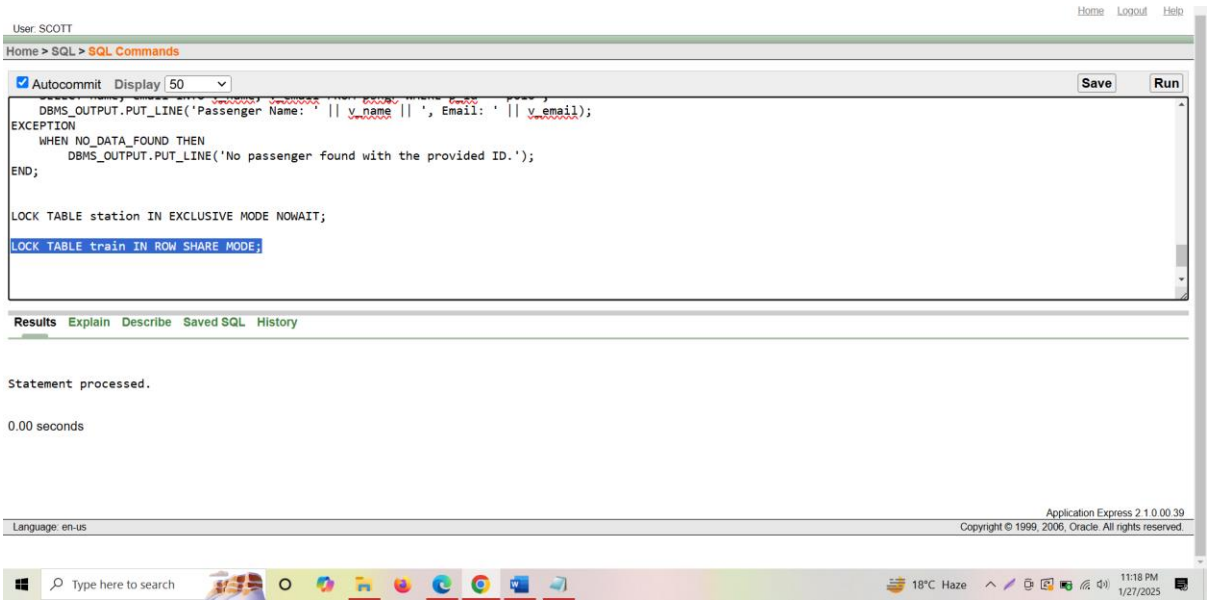
a) How can you lock the entire station table to prevent all modifications, allowing only SELECT operations?

Ans:
LOCK TABLE station IN EXCLUSIVE MODE NOWAIT;



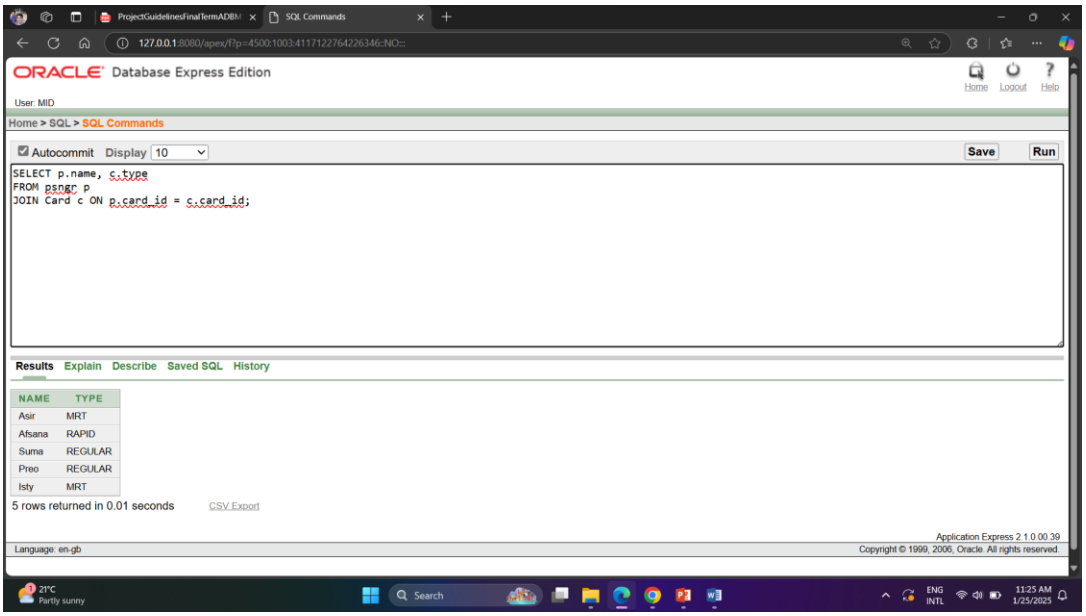
b) How can you allow multiple users to read or update rows in the train table while preventing others from locking the table exclusively?

Ans:
LOCK TABLE train IN ROW SHARE MODE;

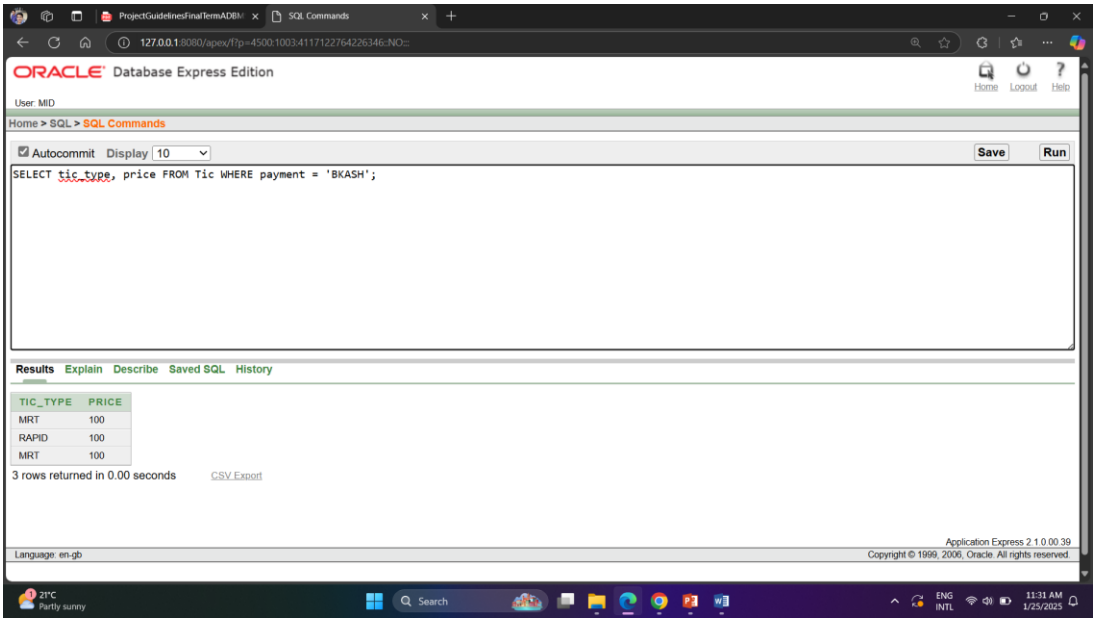


Relational Algebra

a) Retrieve all passenger names and the card type associated with them.
Relational Algebra: $\pi_{\text{name, type}}(\text{psngr} \bowtie \text{Card})$
Query: SELECT p.name, c.type FROM psngr p JOIN Card c ON p.card_id = c.card_id;

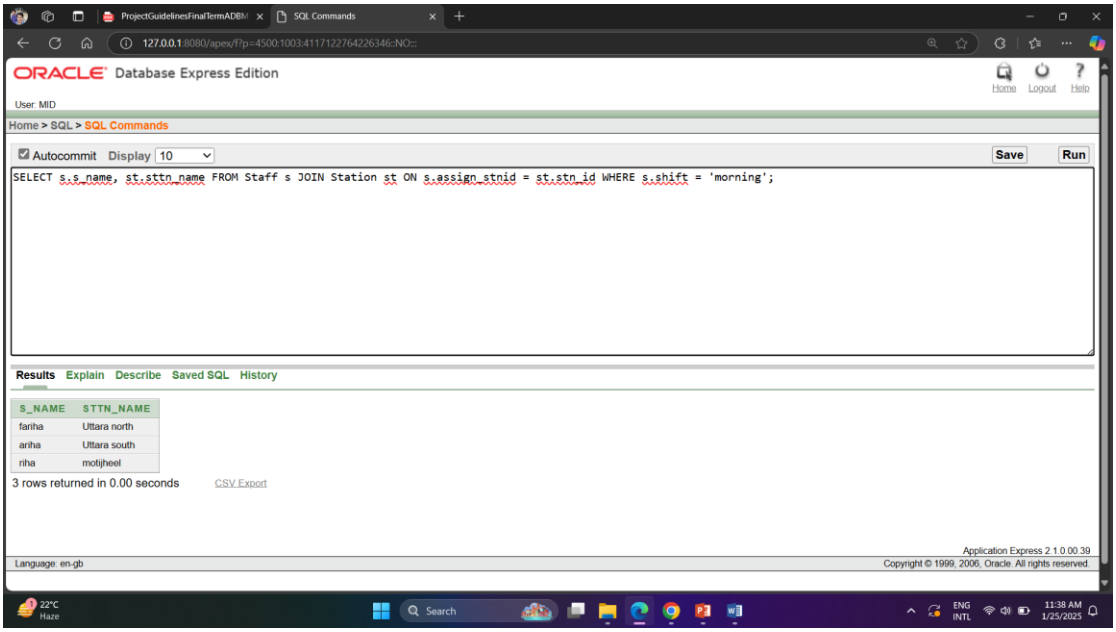


b) Find the ticket type and price for all tickets purchased via 'BKASH'.
Relational Algebra: $\pi_{\text{tic\_type, price}}(\sigma_{\text{payment='BKASH'}}(\text{Tic}))$
Query: SELECT tic_type, price FROM Tic WHERE payment = 'BKASH';

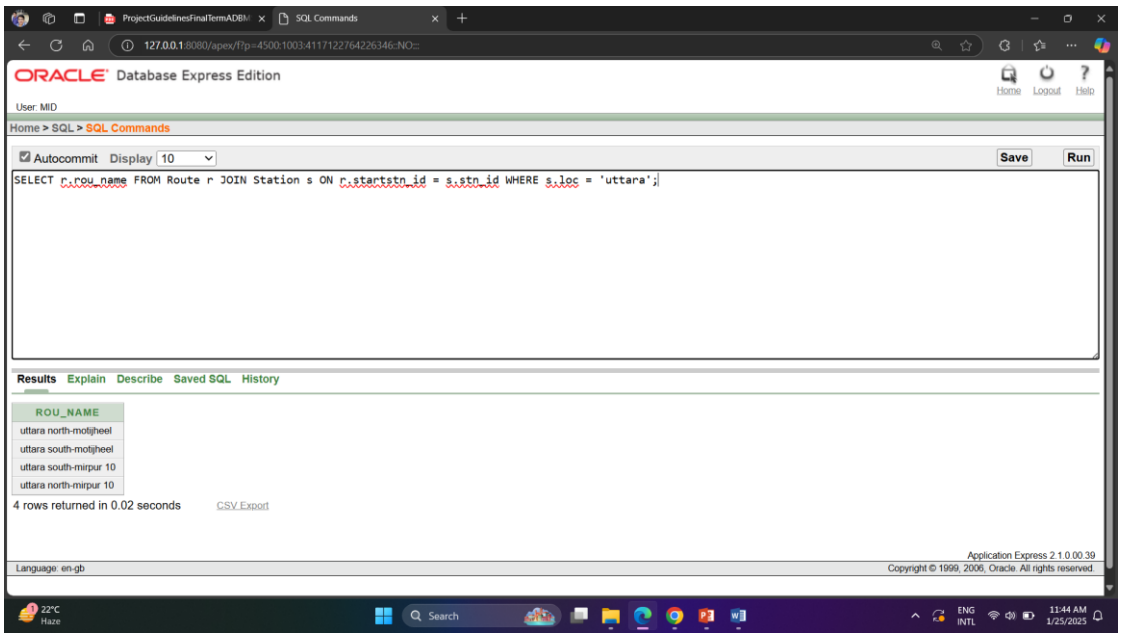


c) List all staff members who work in the 'morning' shift along with their assigned station names.

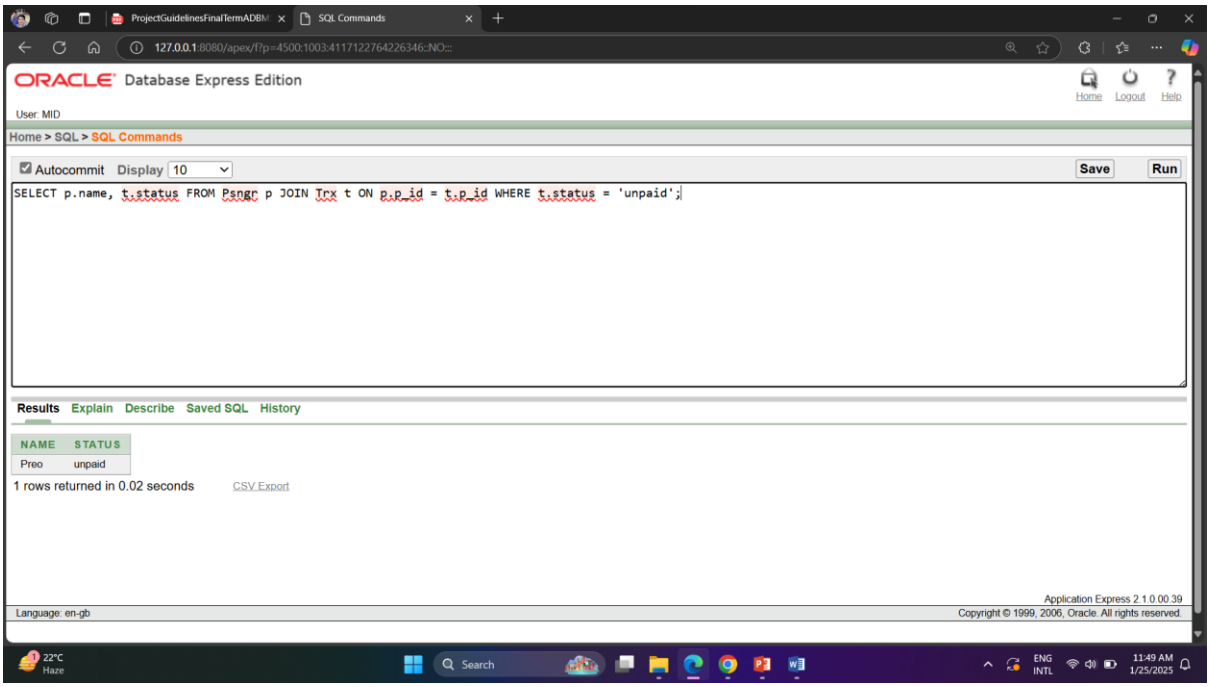
Relational Algebra: $\pi_{s\_name,\ sttn\_name}(\sigma_{shift='morning'} (Staff \bowtie Station))$
Query: SELECT s.s_name, st.sttn_name FROM Staff s JOIN Station st ON s.assign_stnid = st.stn_id WHERE s.shift = 'morning';



d) Retrieve the names of routes that start from stations in 'uttara'.
Relational Algebra: $\pi_{rou\_name}(\sigma_{loc='uttara'} (Station) \bowtie Route)$
Query: SELECT r.rou_name FROM Route r JOIN Station s ON r.startstn_id = s.stn_id WHERE s.loc = 'uttara';



e) Find the passengers who have unpaid transactions.
Relational Algebra: $\pi_{\text{name, status}}(\sigma_{\text{status='unpaid'}} (\text{Trx} \bowtie \text{Psngr}))$
Query: SELECT p.name, t.status FROM Psngr p JOIN Trx t ON p.p_id = t.p_id WHERE t.status = 'unpaid';



Conclusion

The Metro Rail Management System is designed to revolutionize metro operations by delivering a streamlined, efficient, and user-centric platform. It addresses critical operational areas such as ticketing, scheduling, route management, and staff coordination. By integrating real-time updates, intuitive UI design, and robust data management features, the system ensures seamless experience for both passengers and administrators.

The project focuses on providing:

Enhanced passenger convenience in ticketing and journey planning. Operational efficiency for metro administrators through effective management tools. Scalable and accessible system that can be adapted to diverse metro networks. This system bridges the gap between passenger needs and operational demands, offering a sustainable solution to modernize metro station management.