# 

CAR RENTAL DATABASE SYSTEM

Database Project

A logo of a college

AI-generated content may be incorrect.

**Deliverable 3**

**Submitted by**

|  |  |
| --- | --- |
| **Member Name** | **Registration Number** |
| Farah Tanveer | (BCSF24M016) |
| Ayesha Bint E Younas | (BCSF24M032) |
| Gulshan Mushtaq | (BCSF24M046) |
| Hafsa Kokab | (BCSF24M051) |

**Submitted To**

Dr. Irfana Bibi

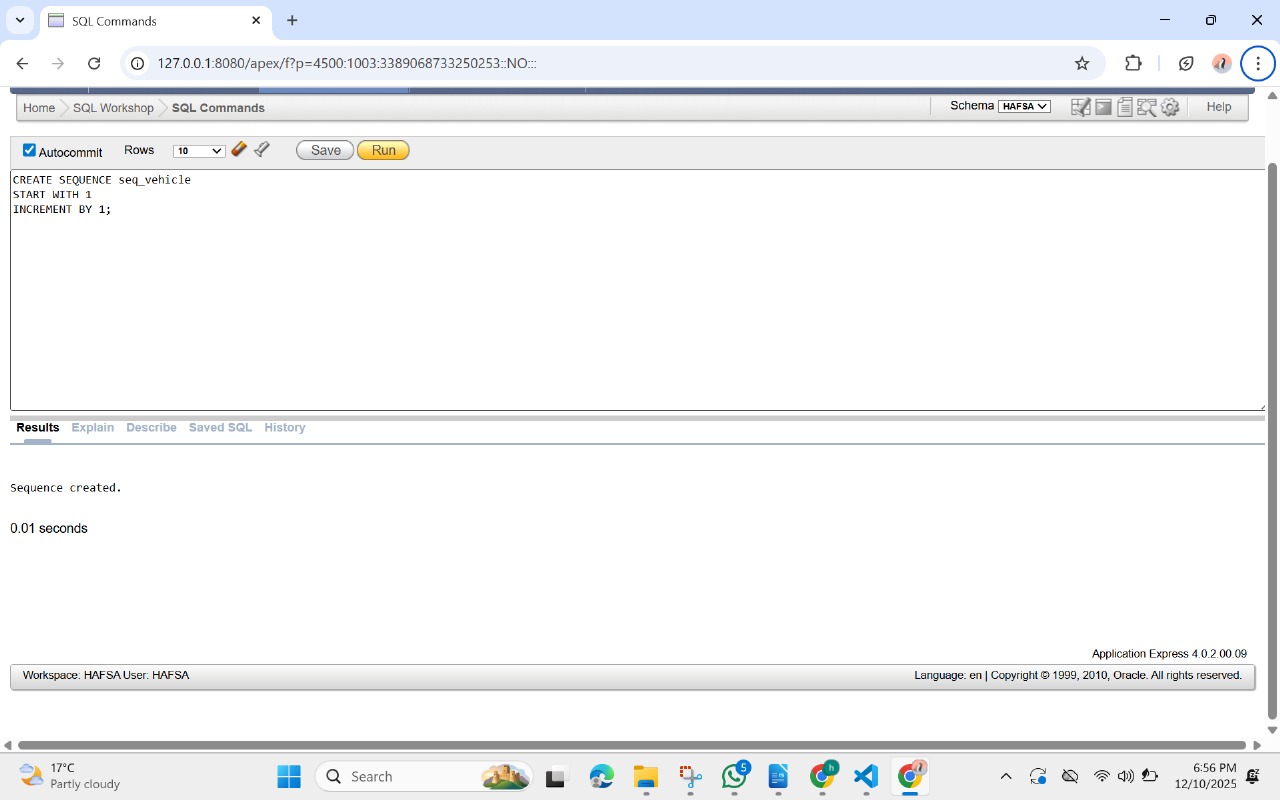
# 

# SEQUENCES

# Vehicle sequence:

CREATE SEQUENCE Seq\_Vehicle

START WITH 1

INCREMENT BY 1;

# Company Sequence:

Create sequence SEQ\_COMPANY

Start with 1

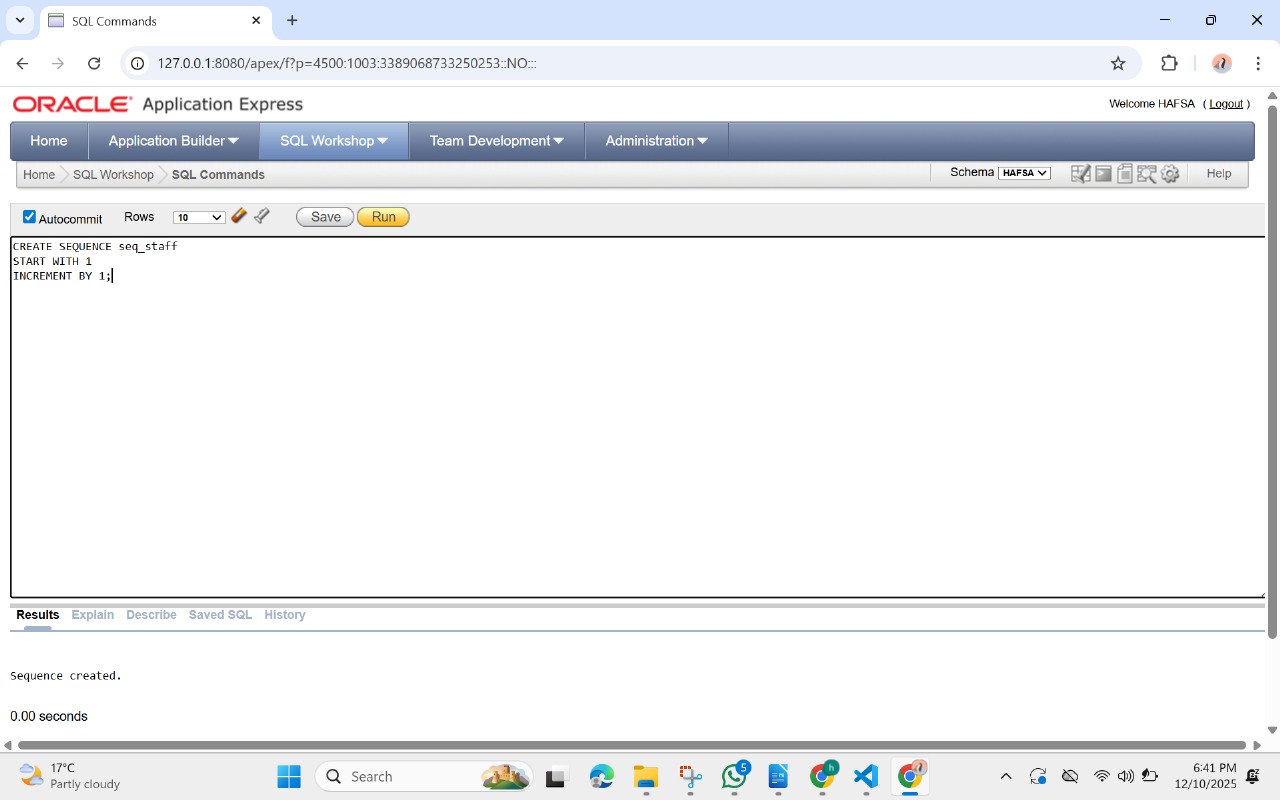
Increment by 1;

# 

# Staff Sequence:

Create sequence seq\_staff

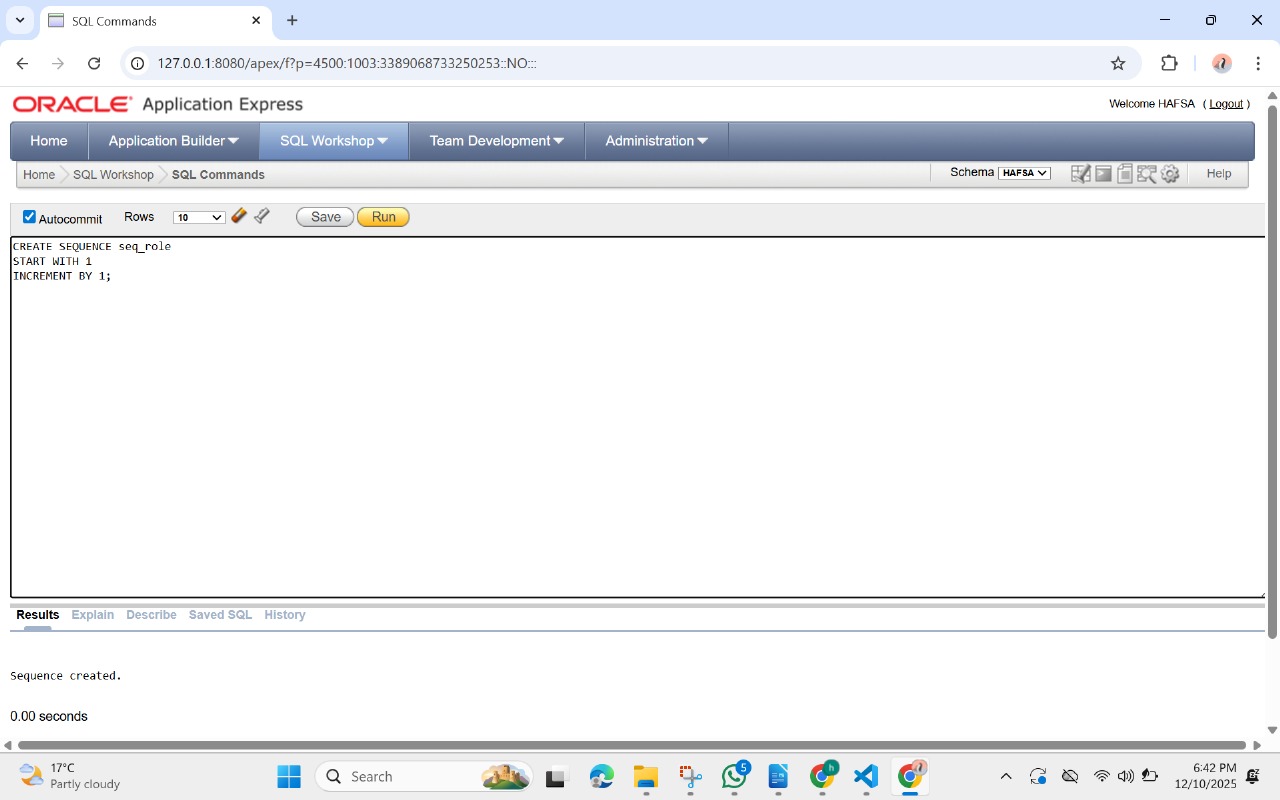
Start with 1

Increment by 1;

# Role Sequence:

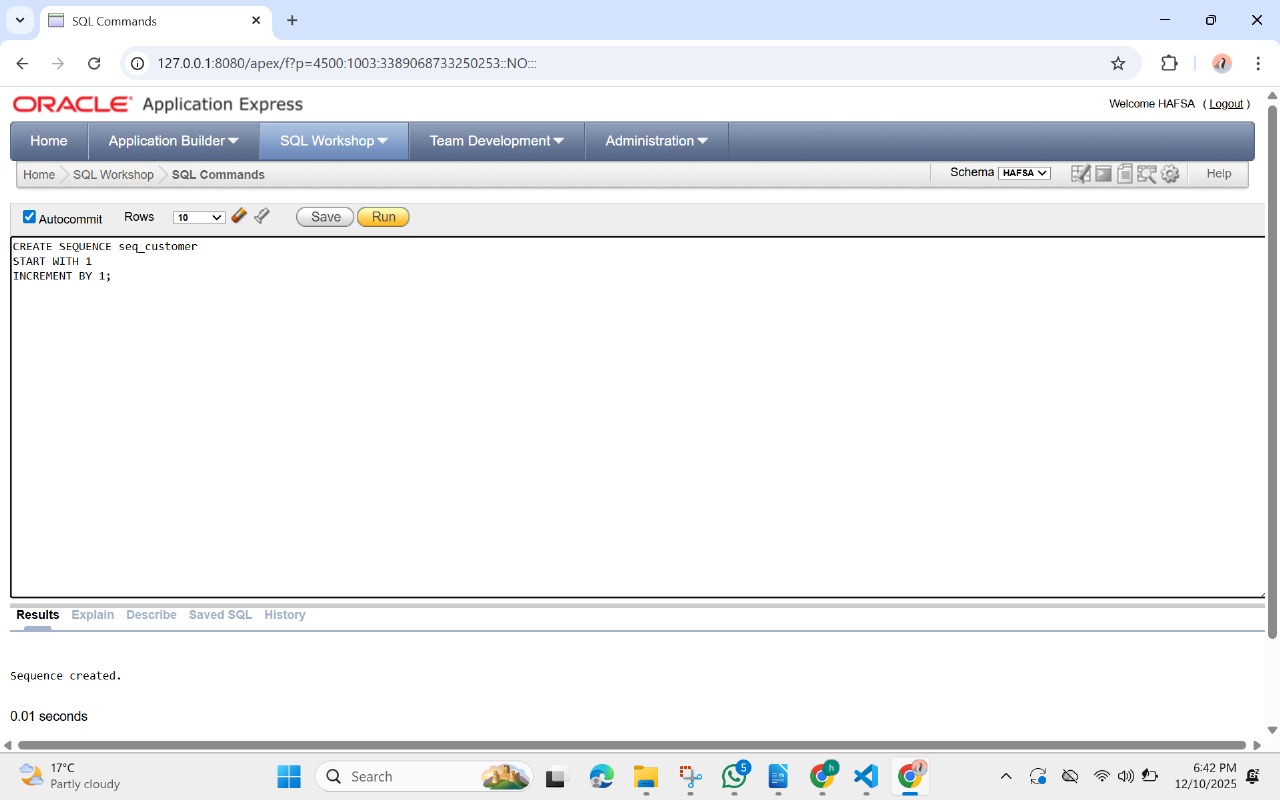
Create sequence seq\_role

Start with 1

Increment by 1;Customer Sequence:

Create sequence seq\_customer

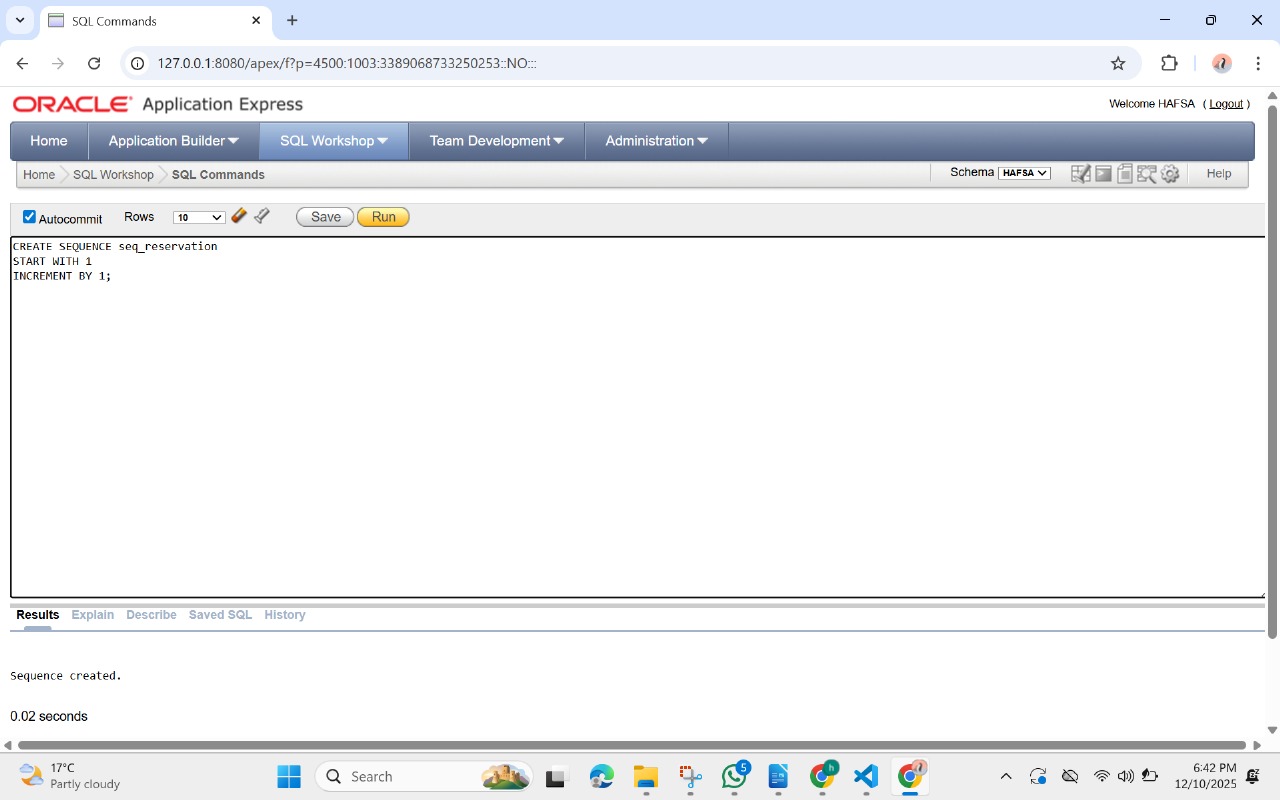
Start with 1

Increment by 1;

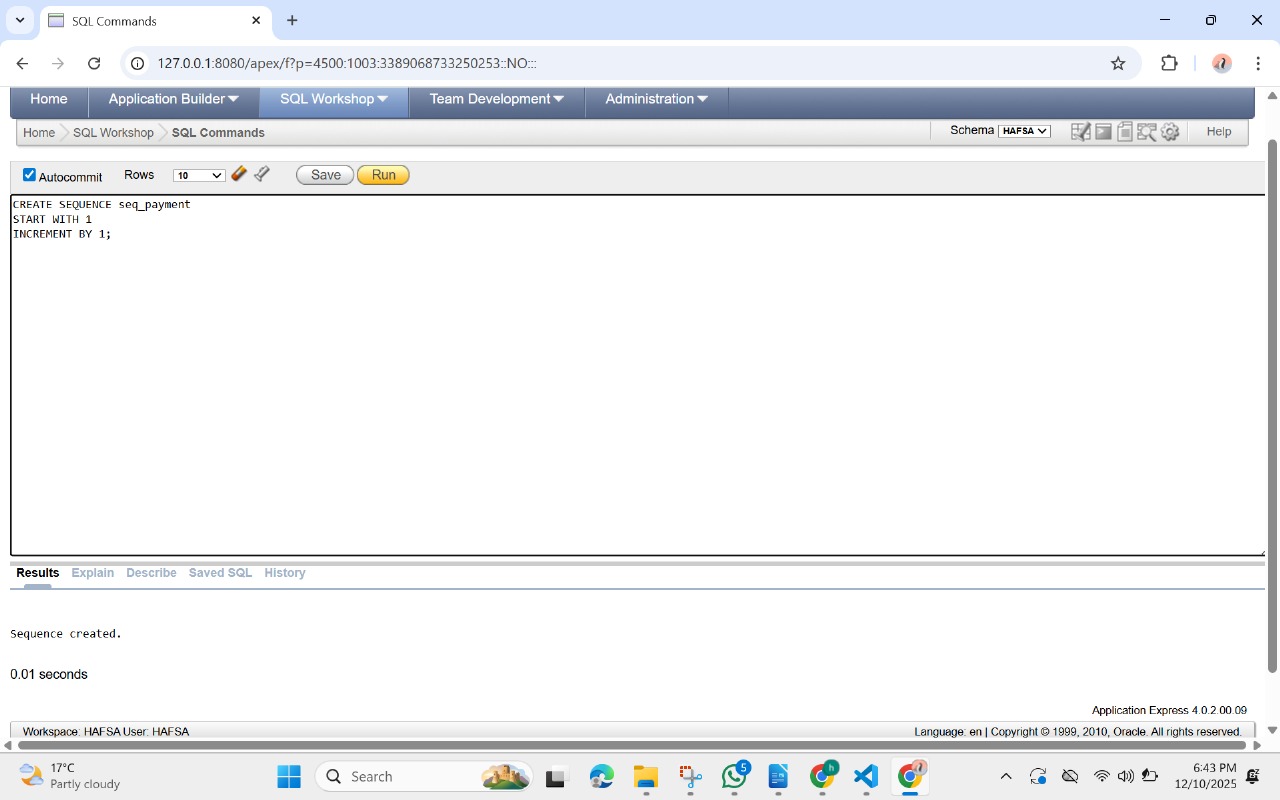
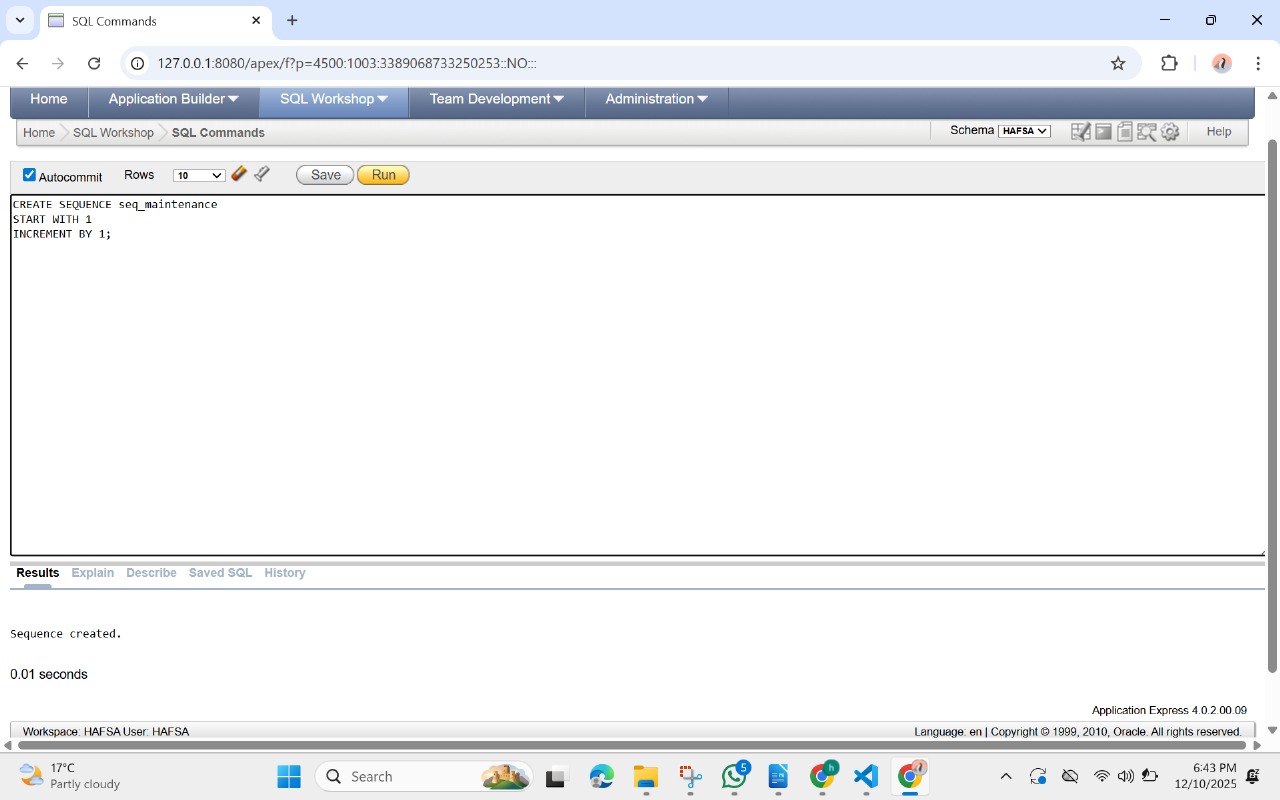
# Reservation Sequence:

CREATE SEQUENCE seq\_reservation

START WITH 1

INCREMENT BY 1;

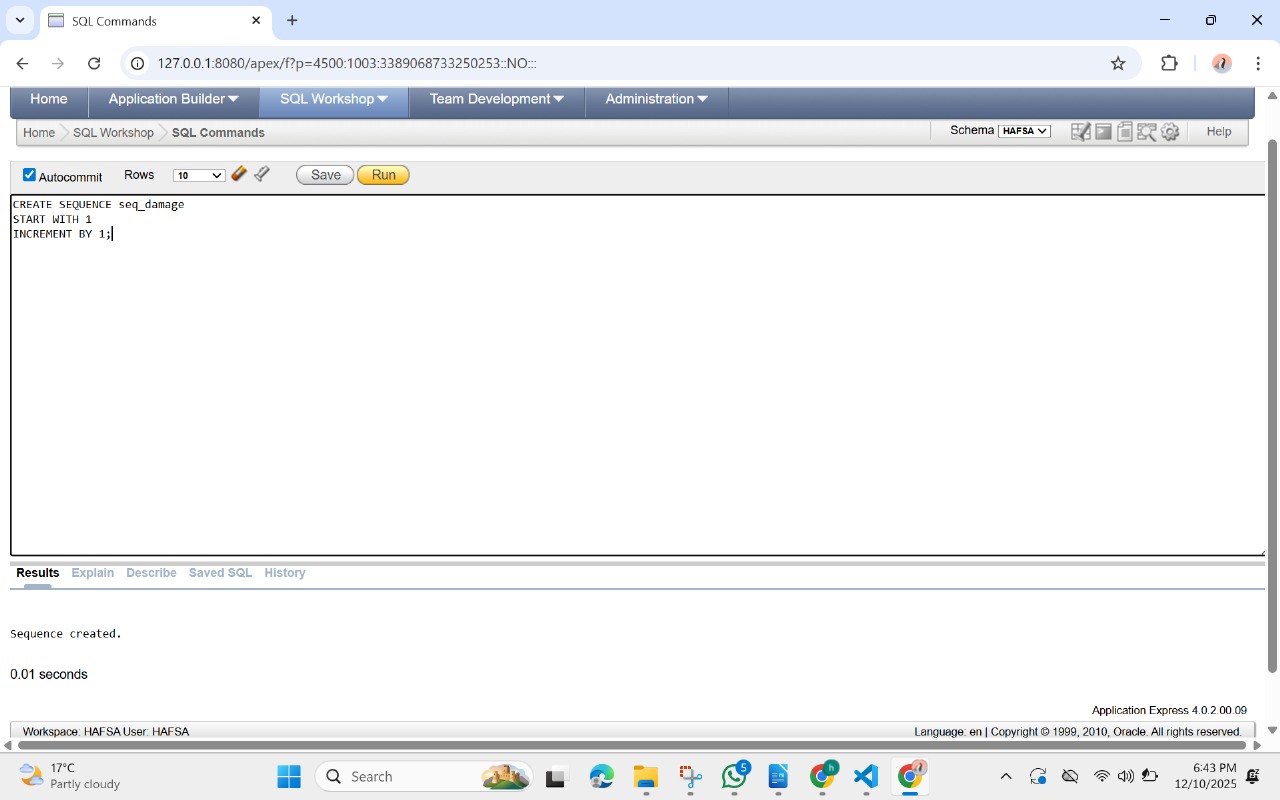
Payment Sequence:  
create sequence SEQ\_PAYMENT  
start with 1  
increment by 1;

Maintenance Sequence:  
create sequence SEQ\_MAINTENANCE  
start with 1  
increment by 1;

# Damage report Sequence:

Create sequence seq\_damage

Start with 1

Increment by 1;

# STORED PROCEDURES

## Add\_Reservation\_Safely

CREATE OR REPLACE PROCEDURE Add\_Reservation\_Safely(

p\_id NUMBER,

p\_vehicle NUMBER,

p\_customer NUMBER,

p\_staff NUMBER,

p\_driver\_required CHAR,

p\_start DATE,

p\_end DATE,

p\_rate NUMBER

)

AS

v\_vehicle\_status VARCHAR2(20);

v\_blacklist CHAR(1);

BEGIN

-- Check vehicle availability

SELECT Status INTO v\_vehicle\_status FROM Vehicle WHERE ID = p\_vehicle;

IF v\_vehicle\_status <> 'Active' THEN

RAISE\_APPLICATION\_ERROR(-20010, 'Vehicle is not available for reservation');

END IF;

-- Check if customer is blacklisted

SELECT BlackListStatus INTO v\_blacklist FROM Customer WHERE ID = p\_customer;

IF v\_blacklist = 'Y' THEN

RAISE\_APPLICATION\_ERROR(-20011, 'Blacklisted customers cannot make reservations');

END IF;

-- Insert reservation

INSERT INTO Reservation (ID, Vehicle\_id, Customer\_id, Staff\_id, DriverRequired,

Rent\_StartDate, Rent\_EndDate, RateNoBooking,

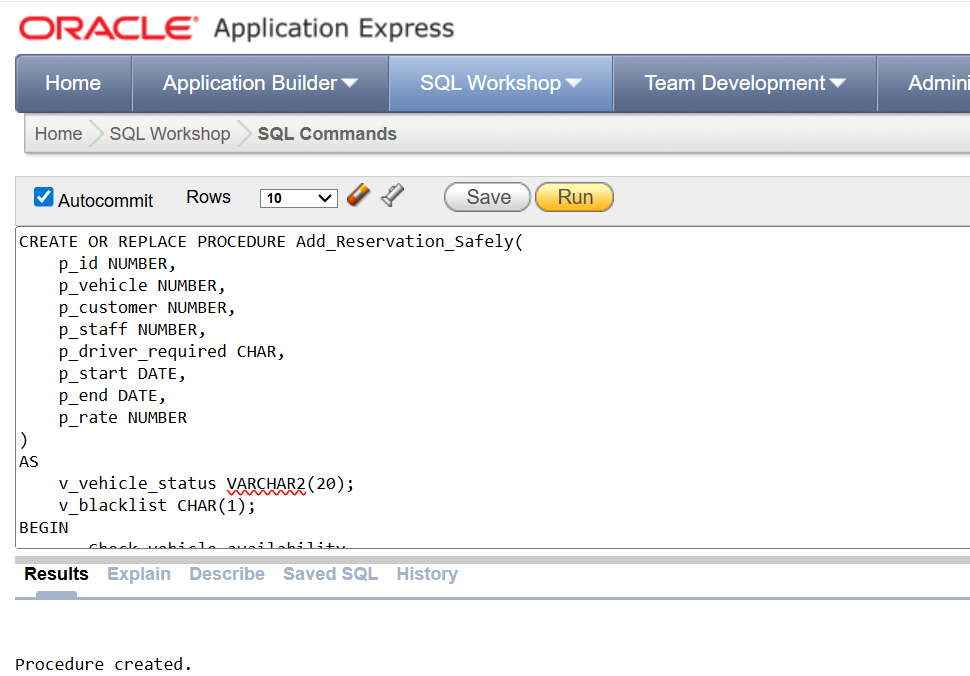
PriceBasis, Status, PaymentStatus)

VALUES (p\_id, p\_vehicle, p\_customer, p\_staff, p\_driver\_required,

p\_start, p\_end, p\_rate, 'Daily', 'Reserved', 'UNPAID');

END;

/



Cancellation of reservation

CREATE OR REPLACE PROCEDURE Cancel\_Reservation

(

p\_reservation\_id NUMBER

)

AS

v\_vehicle NUMBER;

BEGIN

-- 1. Get Vehicle ID of this reservation

SELECT Vehicle\_id

INTO v\_vehicle

FROM Reservation

WHERE ID = p\_reservation\_id;

-- 2. Cancel the reservation

UPDATE Reservation

SET Status = 'Canceled',

PaymentStatus = 'UNPAID'

WHERE ID = p\_reservation\_id;

-- 3. Mark vehicle as Active (available again)

UPDATE Vehicle

SET Status = 'Active'

WHERE ID = v\_vehicle;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

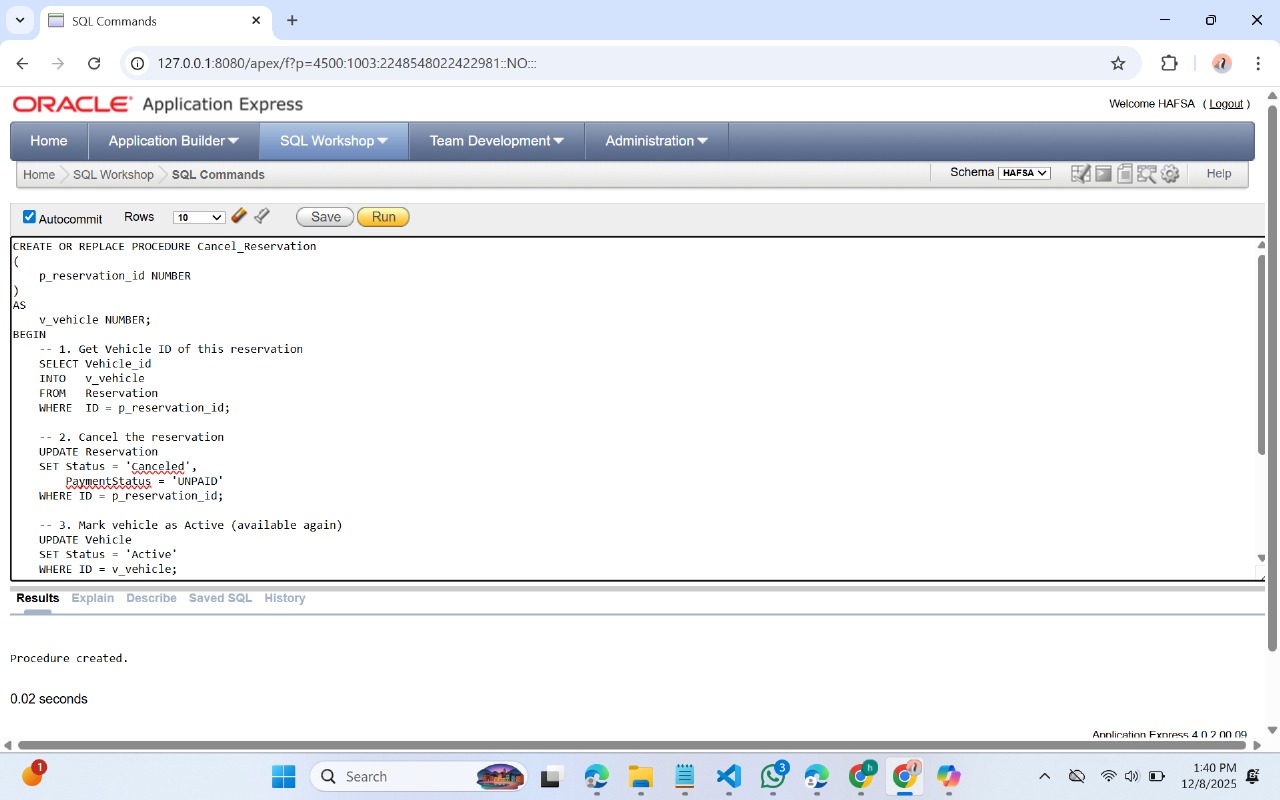
RAISE\_APPLICATION\_ERROR(-20061, 'Cancel failed: Reservation ID not found.');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20062, 'Cancellation error: ' || SQLERRM);

END;

/



TRIGGERS

Update Vehicle Status After Reservation

CREATE OR REPLACE TRIGGER trg\_vehicle\_status\_update

AFTER INSERT OR UPDATE ON Reservation

FOR EACH ROW

BEGIN

IF :NEW.Status = 'Reserved' THEN

UPDATE Vehicle

SET Status = 'Reserved'

WHERE ID = :NEW.Vehicle\_id;

ELSIF :NEW.Status = 'Canceled' THEN

UPDATE Vehicle

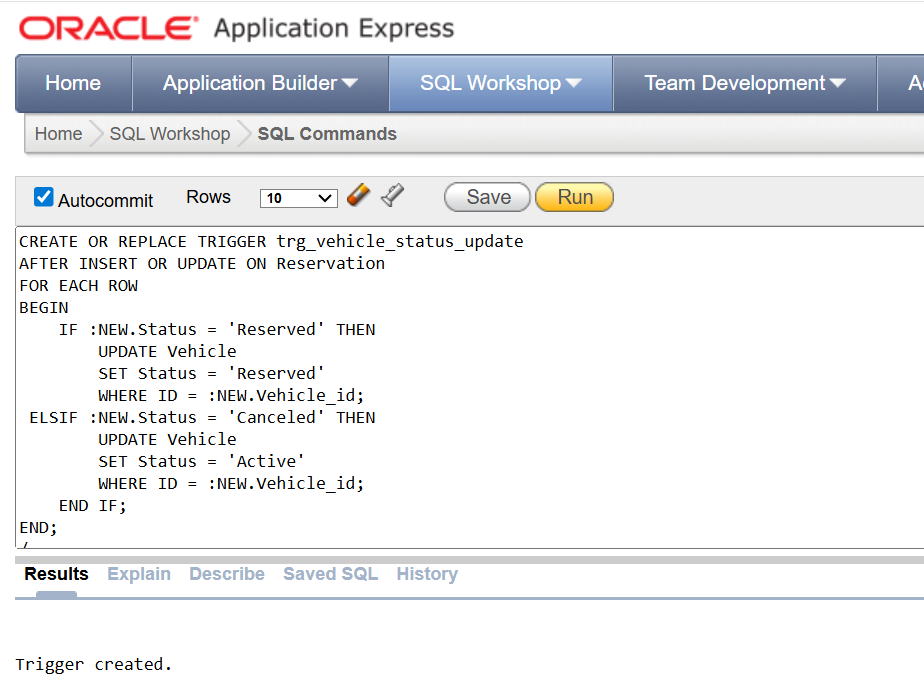
SET Status = 'Active'

WHERE ID = :NEW.Vehicle\_id;

END IF;

END;

/



Auto-Creation of Maintenance on High Damage Report

CREATE OR REPLACE TRIGGER trg\_auto\_maintenance

AFTER INSERT ON Damage\_Report

FOR EACH ROW

WHEN (NEW.SEVERITYLEVEL = 'HIGH')

BEGIN

INSERT INTO Maintenance (ID, Vehicle\_id, Staff\_id, MaintenanceDate, Description, Cost) VALUES (

Maintenance\_seq.NEXTVAL,

:NEW.Vehicle\_id,

2,

SYSDATE,

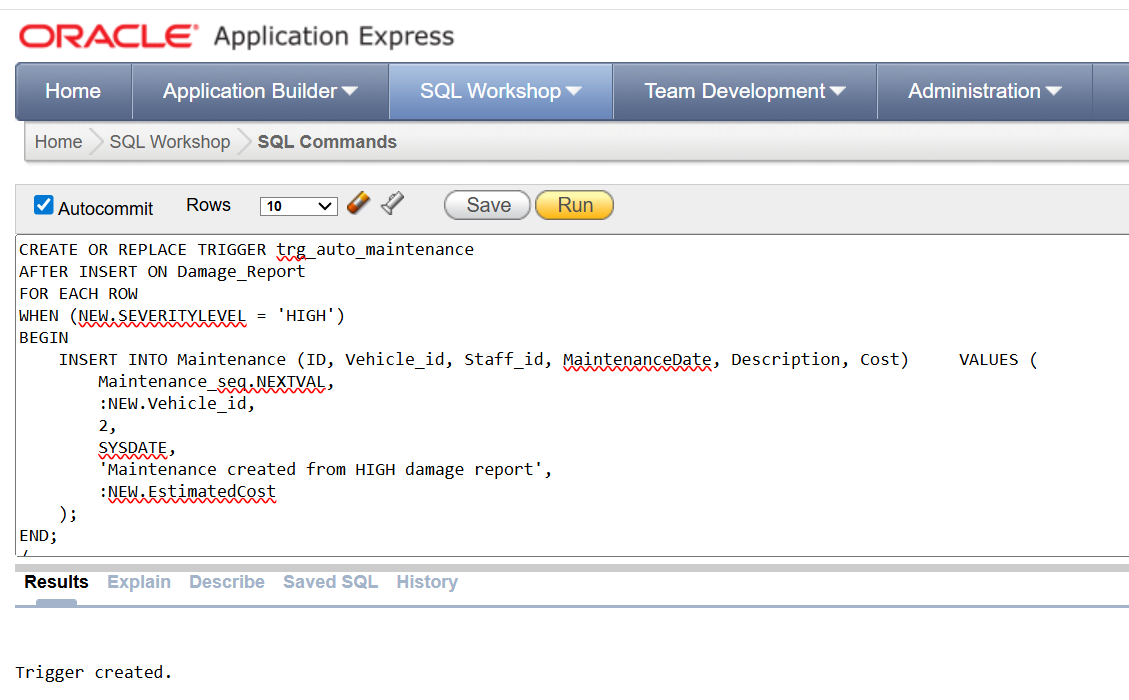
'Maintenance created from HIGH damage report',

:NEW.EstimatedCost

);

END;

/



Prevent overlapping reservations for same vehicle

CREATE OR REPLACE TRIGGER trg\_no\_overlap\_res

BEFORE INSERT OR UPDATE ON Reservation

FOR EACH ROW

DECLARE

v\_count NUMBER;

BEGIN

SELECT COUNT(\*)

INTO v\_count

FROM Reservation

WHERE Vehicle\_id = :NEW.Vehicle\_id

AND Status IN ('Reserved', 'Confirmed')

AND (

:NEW.Rent\_StartDate BETWEEN Rent\_StartDate AND Rent\_EndDate

OR :NEW.Rent\_EndDate BETWEEN Rent\_StartDate AND Rent\_EndDate

OR ( :NEW.Rent\_StartDate <= Rent\_StartDate AND :NEW.Rent\_EndDate >= Rent\_EndDate )

);

IF v\_count > 0 THEN

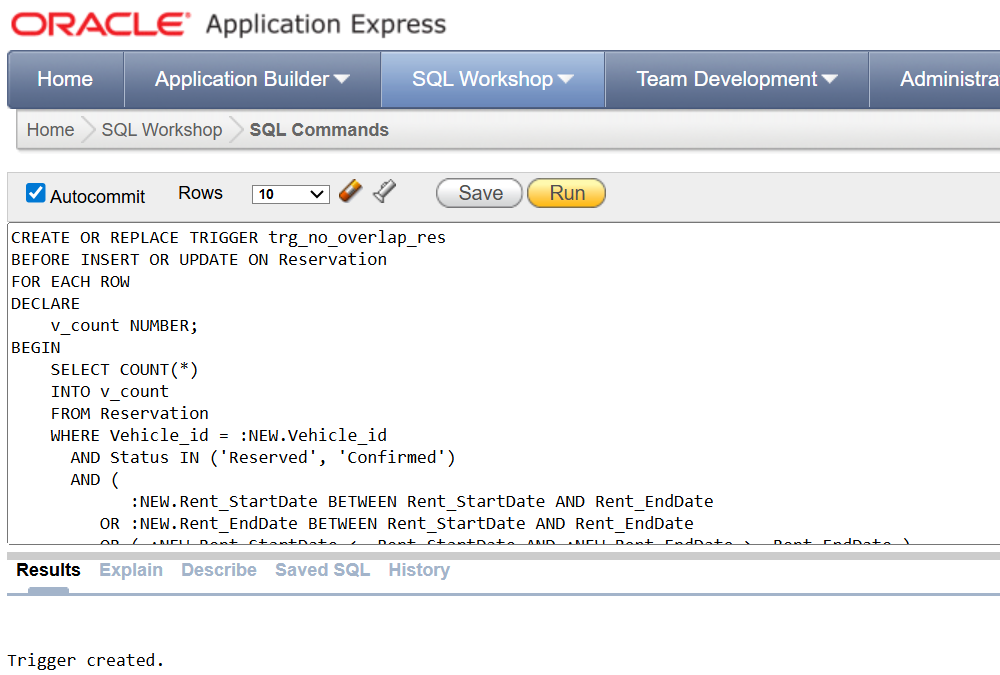
RAISE\_APPLICATION\_ERROR(-20002,

'This vehicle already has an active reservation in this date range.');

END IF;

END;

/



No reservations can be made for under maintenance vehicle

CREATE OR REPLACE TRIGGER trg\_res\_veh\_main

BEFORE INSERT OR UPDATE ON Reservation

FOR EACH ROW

DECLARE

v\_status VARCHAR2(20);

BEGIN

SELECT Status INTO v\_status

FROM Vehicle

WHERE ID = :NEW.Vehicle\_id;

IF v\_status = 'MAINTENANCE' THEN

RAISE\_APPLICATION\_ERROR(

-30004,

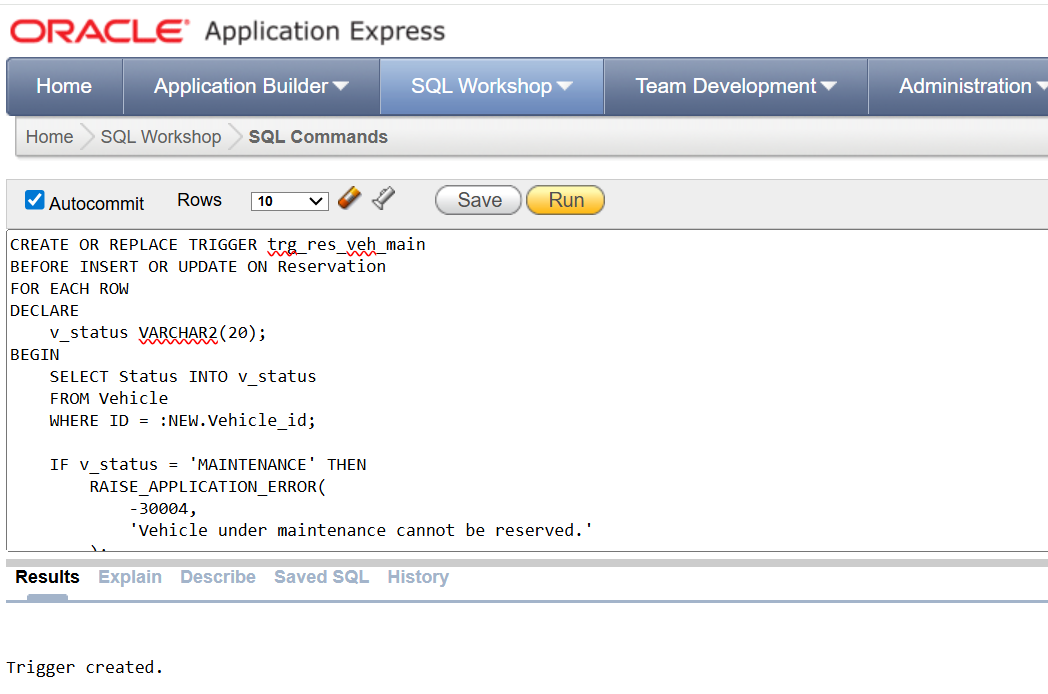
'Vehicle under maintenance cannot be reserved.'

);

END IF;

END;

/



No maintenance can be made for reserved vehicle

CREATE OR REPLACE TRIGGER trg\_no\_maintenance\_if\_reserved

BEFORE INSERT ON Maintenance

FOR EACH ROW

DECLARE

v\_status VARCHAR2(20);

BEGIN

SELECT Status INTO v\_status

FROM Vehicle

WHERE ID = :NEW.Vehicle\_id;

IF v\_status IN ('Reserved', 'Rented') THEN

RAISE\_APPLICATION\_ERROR(

-20003,

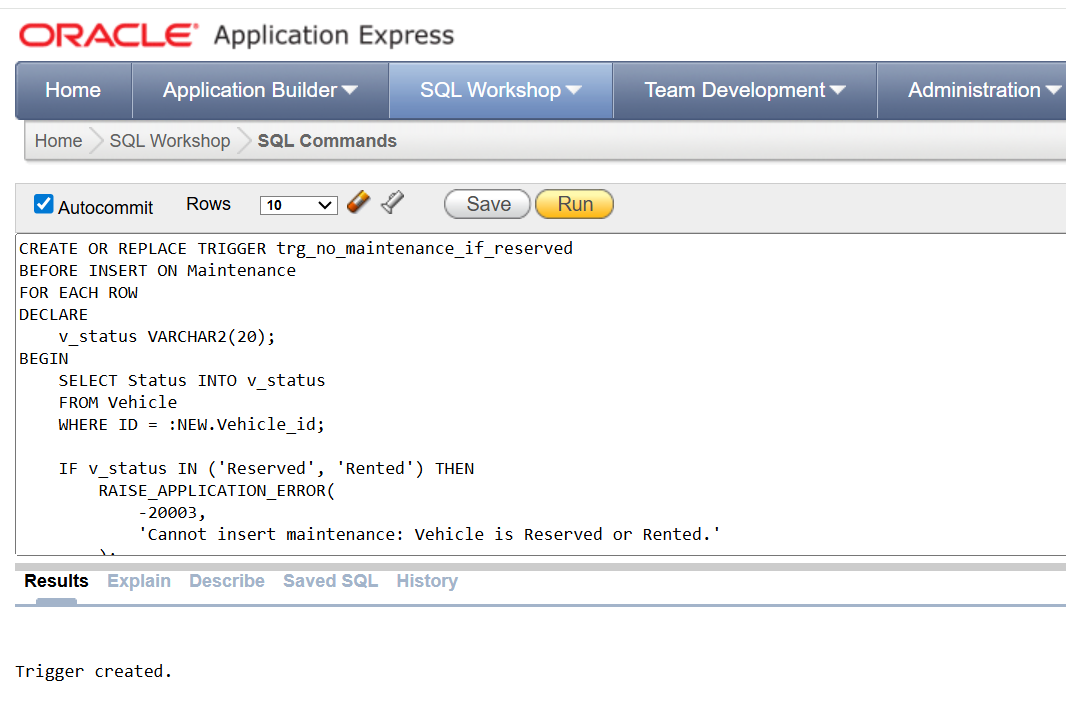
'Cannot insert maintenance: Vehicle is Reserved or Rented.'

);

END IF;

END;

/



Enforce staff role for maintenance (only mechanics and other staff[cleaner] allowed

CREATE OR REPLACE TRIGGER trg\_mech\_req\_main

BEFORE INSERT ON Maintenance

FOR EACH ROW

DECLARE

v\_role VARCHAR2(30);

BEGIN

SELECT r.Name

INTO v\_role

FROM Staff s

JOIN Role r ON s.Role\_id = r.ID

WHERE s.ID = :NEW.Staff\_id;

IF v\_role <> 'MECHANIC' AND v\_role <> 'Other Staff' THEN

RAISE\_APPLICATION\_ERROR(

-20004,

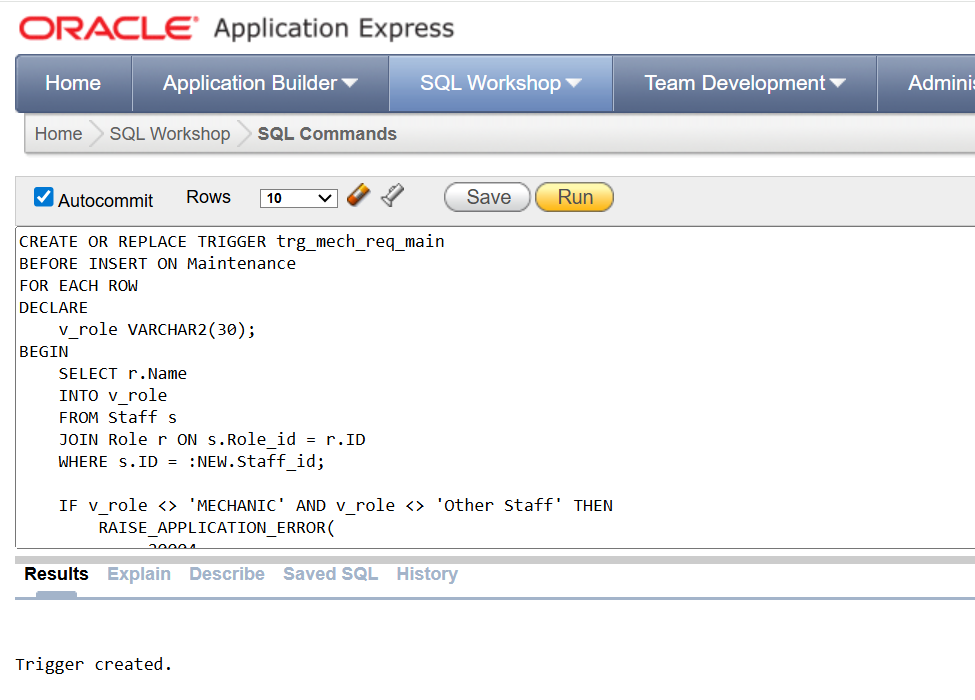
'Only mechanics (Other Staff role) can perform maintenance.'

);

END IF;

END;

/



Prevent blacklisted customer from making reservations

CREATE OR REPLACE TRIGGER trg\_blklist\_res

BEFORE INSERT ON Reservation

FOR EACH ROW

DECLARE

v\_black CHAR(1);

BEGIN

SELECT BlackListStatus

INTO v\_black

FROM Customer

WHERE ID = :NEW.Customer\_id;

IF v\_black = 'Y' THEN

RAISE\_APPLICATION\_ERROR(

-20001,

'This customer is blacklisted and cannot make a reservation.'

);

END IF;

END;

/

## CALCULATING PAYMENT FOR RESERVATION CREATE OR REPLACE TRIGGER trg\_calculate\_payment

BEFORE INSERT ON Payment

FOR EACH ROW

DECLARE

v\_rate NUMBER;

v\_start DATE;

v\_end DATE;

v\_days NUMBER;

v\_amount NUMBER;

v\_customer NUMBER;

BEGIN

-- Fetch Reservation Data

SELECT r.RateNoBooking,

r.Rent\_StartDate,

r.Rent\_EndDate,

r.Customer\_id

INTO v\_rate, v\_start, v\_end, v\_customer

FROM Reservation r

WHERE r.ID = :NEW.Reservation\_id;

-- Billing Days Calculation

v\_days := CASE

WHEN v\_end = v\_start THEN 1

ELSE CEIL(v\_end - v\_start)

END;

-- Payment amount calculation

v\_amount := v\_rate \* v\_days;

-- Override user input with correct values

:NEW.Amount := v\_amount;

:NEW.Customer\_id := v\_customer;

:NEW.PaymentDate := SYSDATE;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20051,

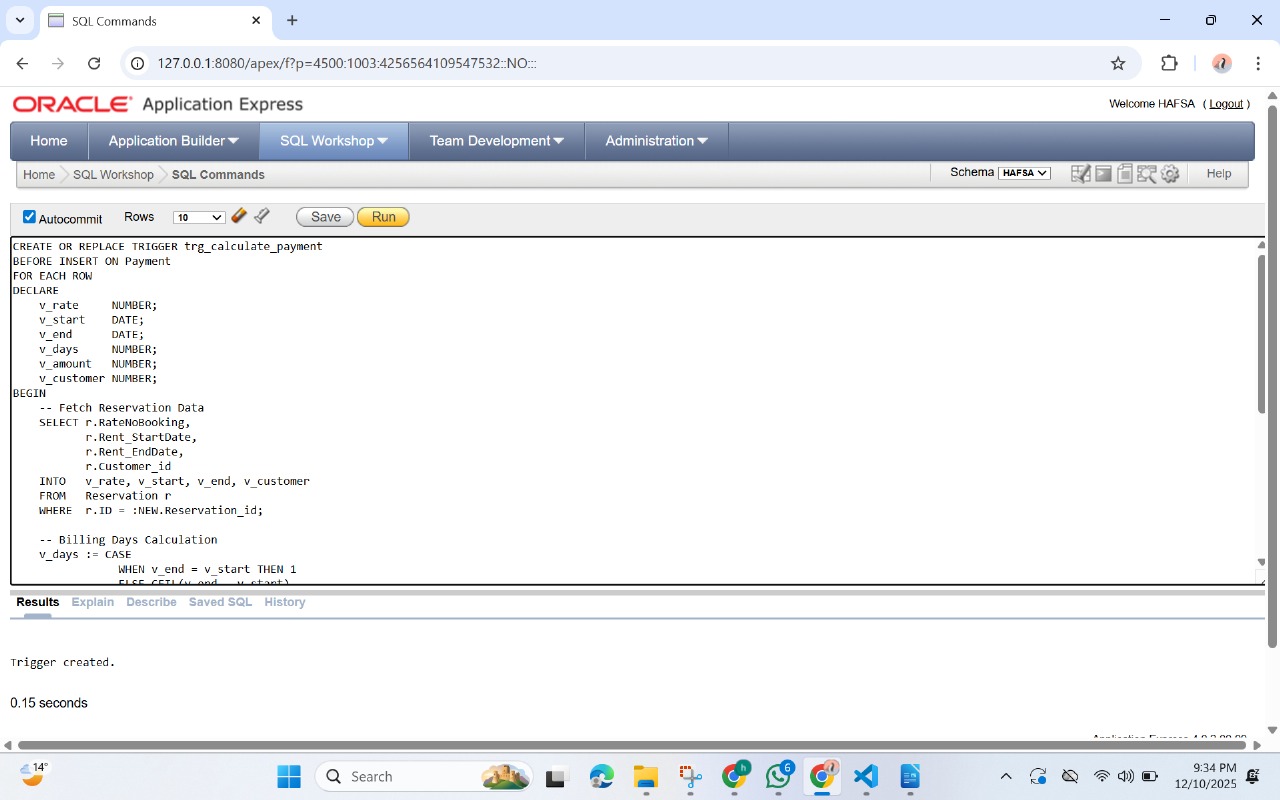
'Reservation not found OR RateNoBooking is NULL.');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20052,

'Payment calculation failed: ' || SQLERRM);

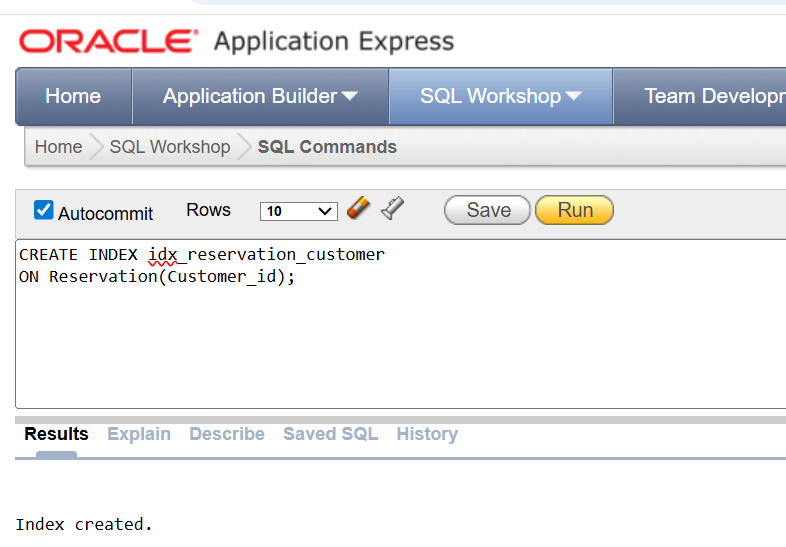
END;

/

# INDEXES

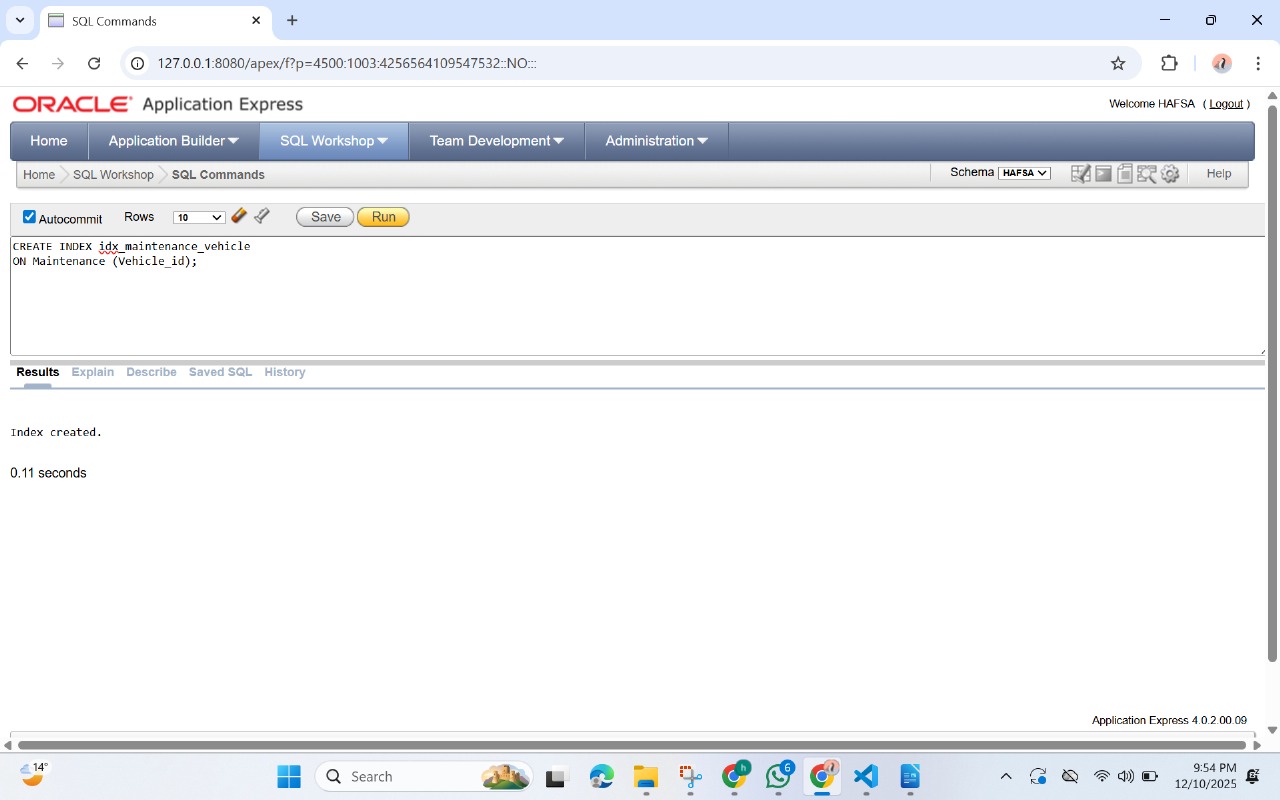
Reservation Customer Index

CREATE INDEX idx\_reservation\_customer

ON Reservation(Customer\_id); 

Creating an index on maintenance table

CREATE INDEX idx\_maintenance\_vehicle  
ON Maintenance (Vehicle\_id);



# VIEWS

View: Customer Summary

CREATE OR REPLACE VIEW vw\_customer\_summary AS

SELECT

c.ID AS CustomerID,

c.FirstName || ' ' || c.LastName AS CustomerName,

c.PhoneNumber,

c.Email,

c.Address,

c.BlackListStatus,

r.ID AS ReservationID,

r.Rent\_StartDate,

r.Rent\_EndDate,

r.Status AS ReservationStatus,

p.Amount AS LastPaymentAmount,

p.PaymentDate AS LastPaymentDate

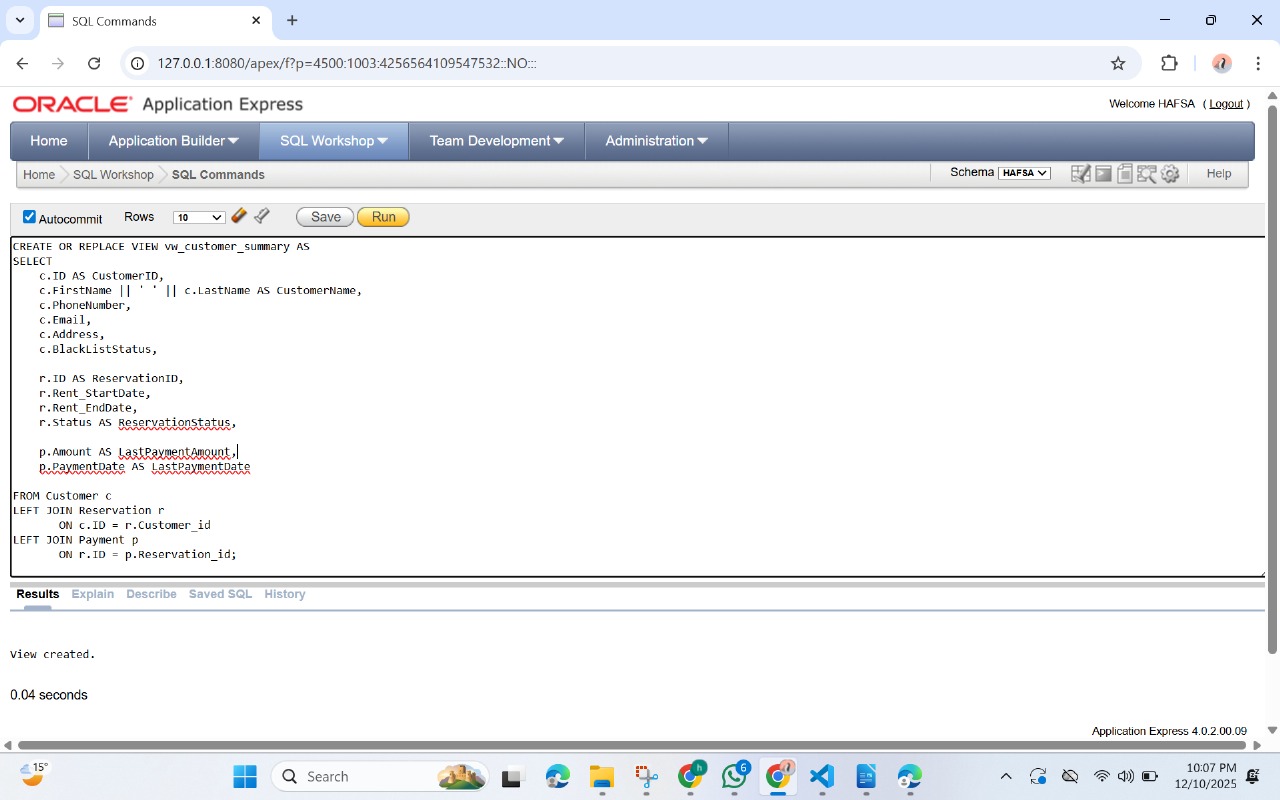
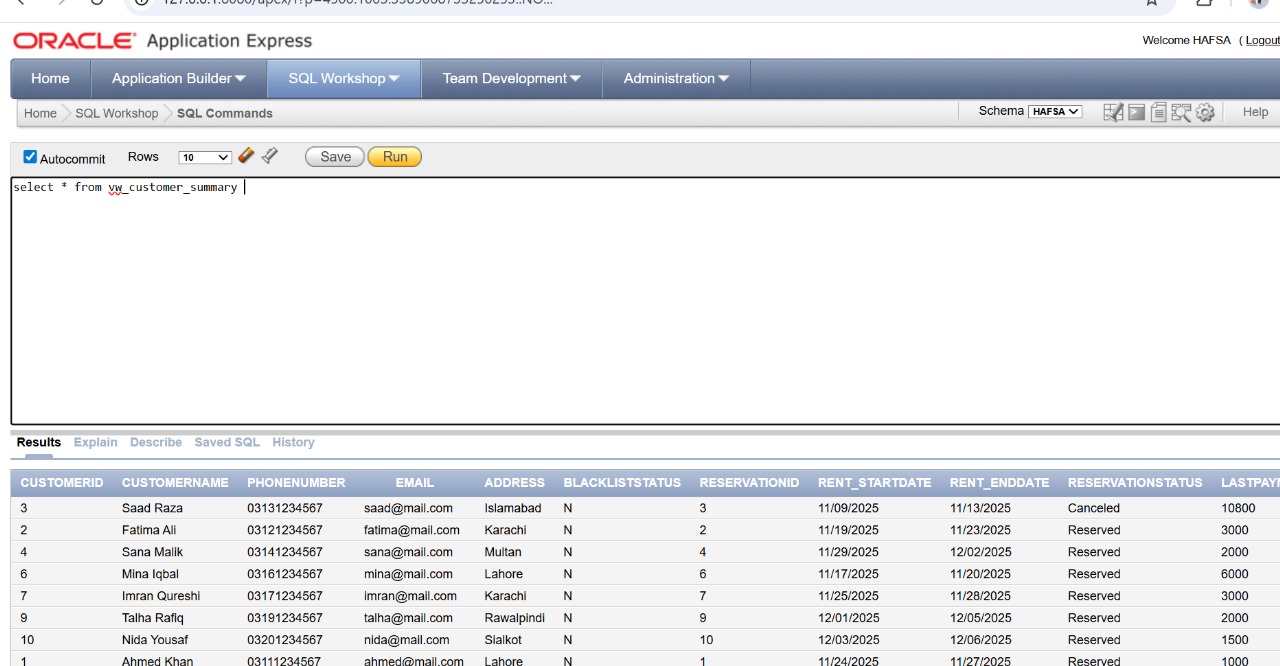
FROM Customer c

LEFT JOIN Reservation r

ON c.ID = r.Customer\_id

LEFT JOIN Payment p

ON r.ID = p.Reservation\_id;



View: Vehicle Damage & Maintenance

CREATE OR REPLACE VIEW vw\_vehicle\_damage\_maintenance AS SELECT d.Vehicle\_id,

d.SeverityLevel,

d.Description AS DamageDetails,

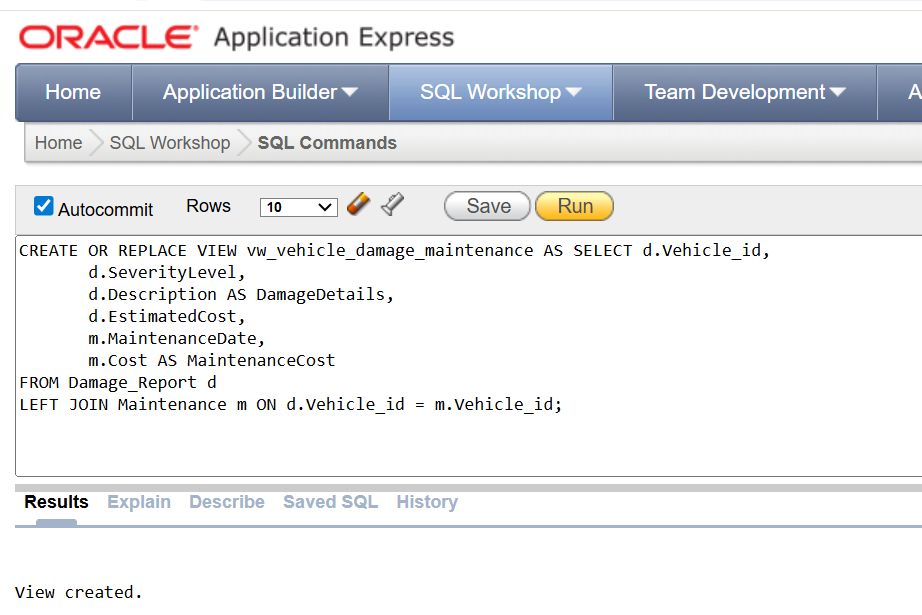
d.EstimatedCost,

m.MaintenanceDate,

m.Cost AS MaintenanceCost

FROM Damage\_Report d

LEFT JOIN Maintenance m ON d.Vehicle\_id = m.Vehicle\_id;

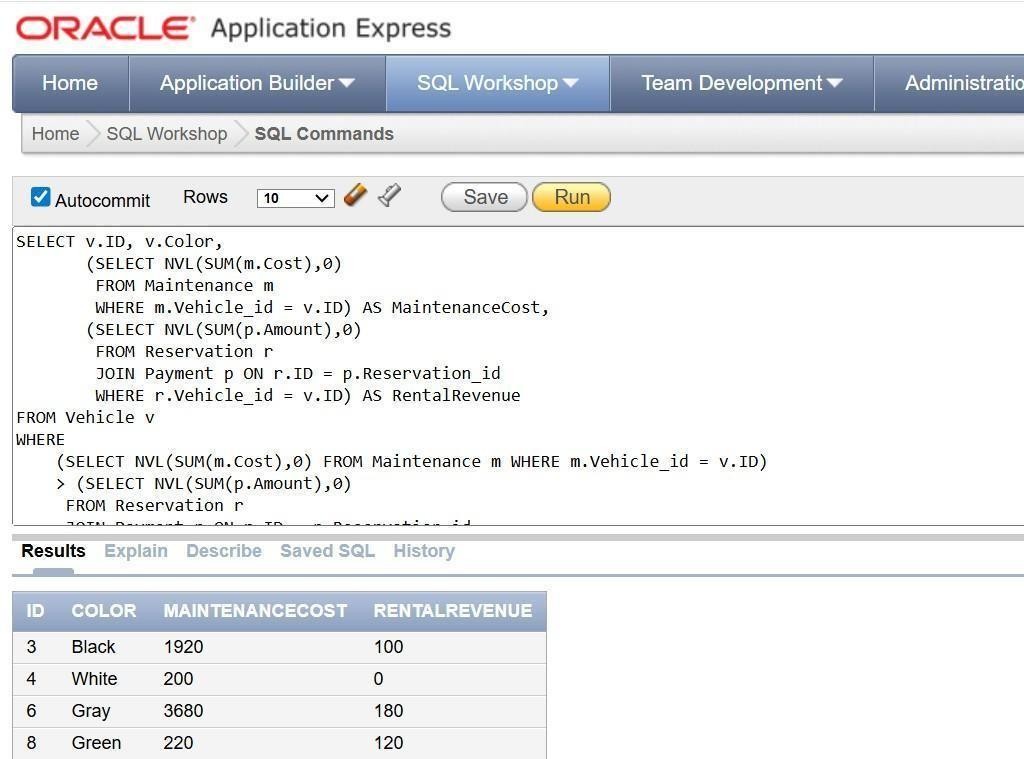


COMPLEX SQL QUEIES

1. **Which vehicles cost more in maintenance than they earned in rental revenue (loss-making vehicles).**

SQL Query

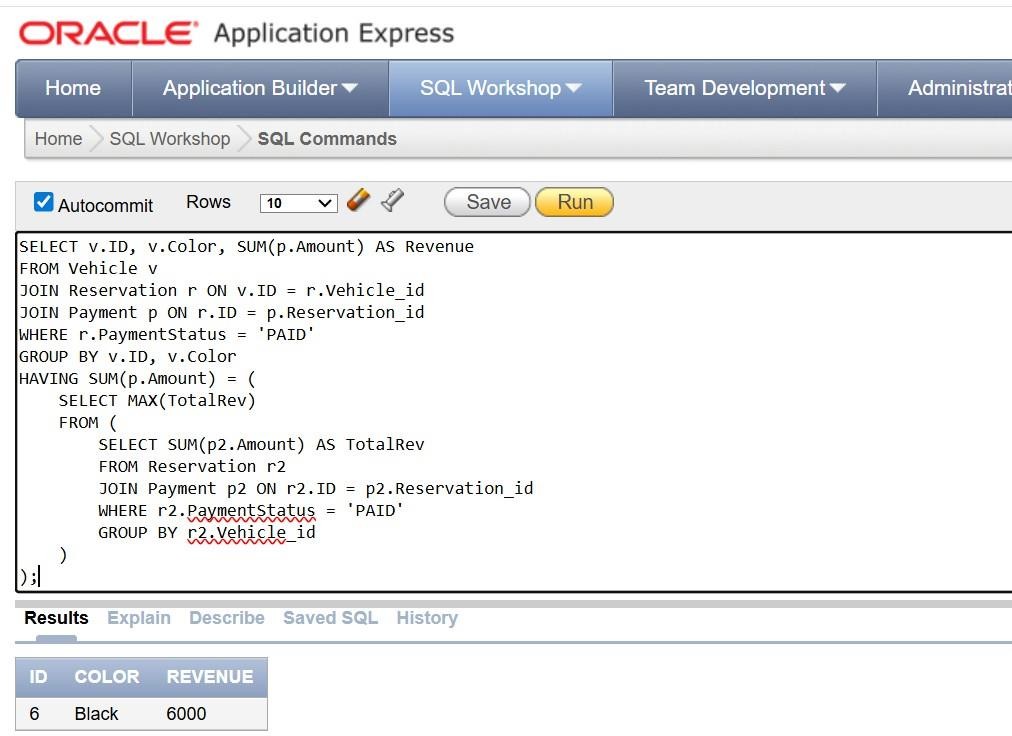
SELECT v.ID, v.Color, (SELECT NVL(SUM(m.Cost),0) FROM Maintenance m WHERE m.Vehicle\_id = v.ID) AS MaintenanceCost, (SELECT NVL(SUM(p.Amount),0) FROM Reservation r JOIN Payment p ON r.ID = p.Reservation\_id WHERE r.Vehicle\_id = v.ID) AS RentalRevenue FROM Vehicle v WHERE (SELECT NVL(SUM(m.Cost),0) FROM Maintenance m WHERE m.Vehicle\_id = v.ID) > (SELECT NVL(SUM(p.Amount),0) FROM Reservation r JOIN Payment p ON r.ID = p.Reservation\_id WHERE r.Vehicle\_id = v.ID)



**2.Which vehicles generated the highest rental revenue from fully paid reservations**.

**SQL Query**

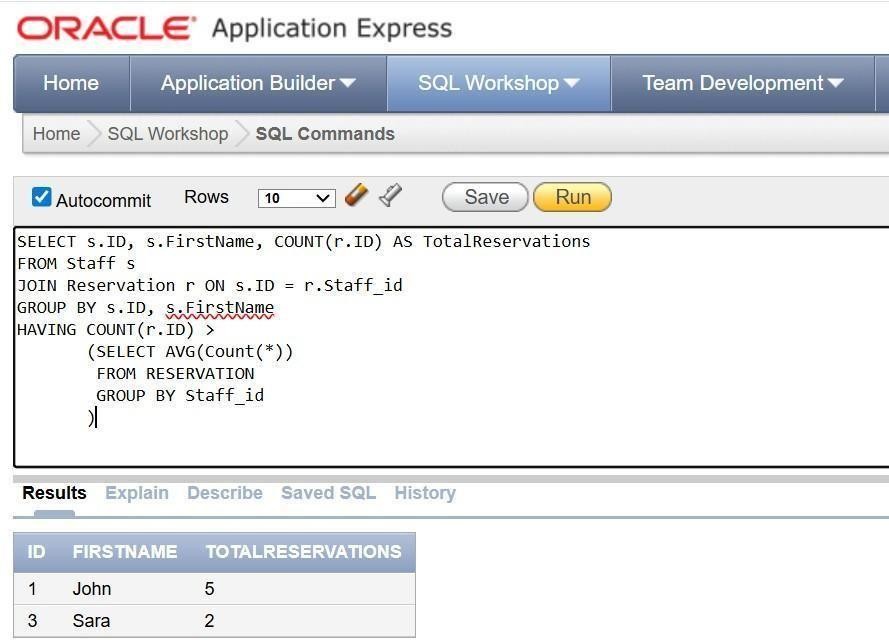
SELECT v.ID, v.Color, SUM(p.Amount) AS Revenue FROM Vehicle v JOIN Reservation r ON v.ID = r.Vehicle\_id JOIN Payment p ON r.ID = p.Reservation\_id WHERE r.PaymentStatus = 'PAID' GROUP BY v.ID, v.Color HAVING SUM(p.Amount) = ( SELECT MAX(TotalRev) FROM ( SELECT SUM(p2.Amount) AS TotalRev FROM Reservation r2 JOIN Payment p2 ON r2.ID = p2.Reservation\_id GROUP BY r2.Vehicle\_id ) )



3.**Which staff members handled more reservations than the average staff workload**

**SQL QUERY**

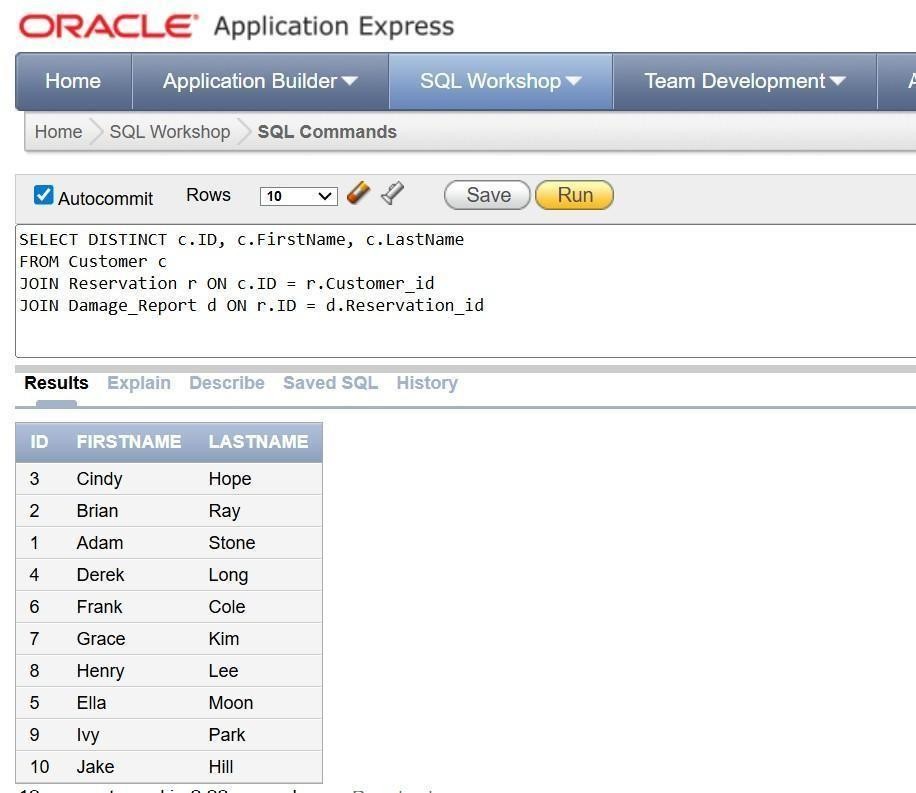
SELECT s.ID, s.FirstName, COUNT(r.ID) AS TotalReservations FROM Staff s JOIN Reservation r ON s.ID = r.Staff\_id GROUP BY s.ID, s.FirstName HAVING COUNT(r.ID) > (SELECT AVG(Count(\*)) FROM RESERVATION GROUP BY Staff\_id )



4. **Which customers rented vehicles that were later reported as damaged.**

**SQL QUERY**

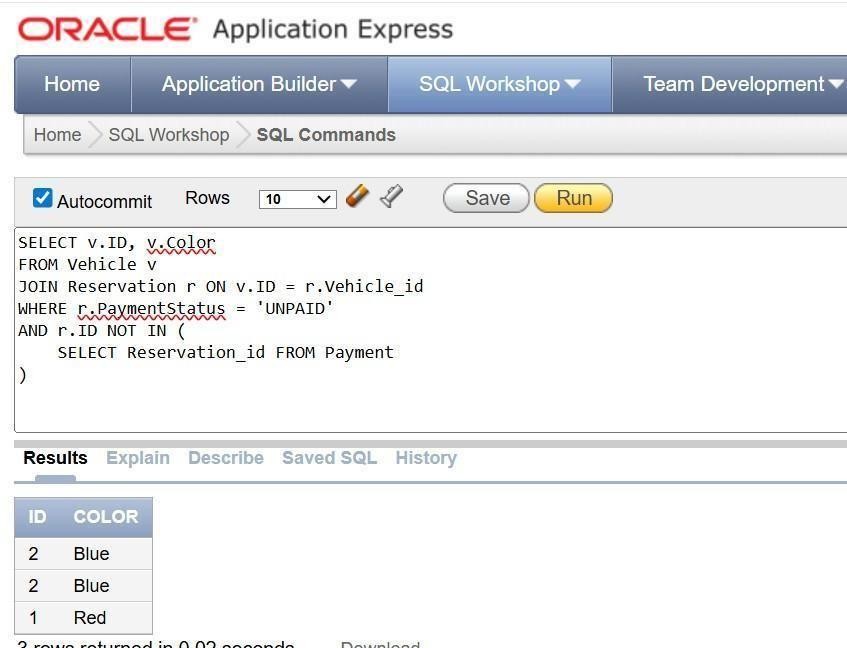
SELECT DISTINCT c.ID, c.FirstName, c.LastName FROM Customer c JOIN Reservation r ON c.ID = r.Customer\_id JOIN Damage\_Report d ON r.ID = d.Reservation\_id



5. **. Which vehicles are currently reserved but have no recorded payments.**

**SQL Query**

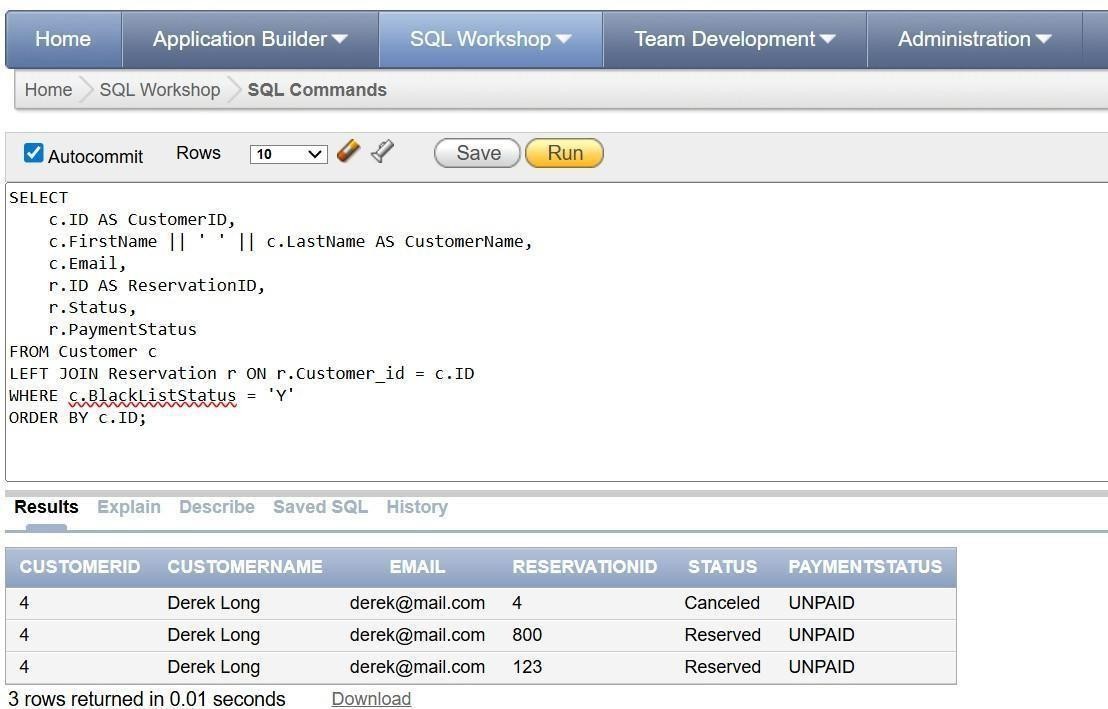
SELECT v.ID, v.Color FROM Vehicle v JOIN Reservation r ON v.ID = r.Vehicle\_id WHERE r.PaymentStatus = 'UNPAID' AND r.ID NOT IN ( SELECT Reservation\_id FROM Payment )



6Show All Blacklisted Customers & Their Reservation History

SQL Query:

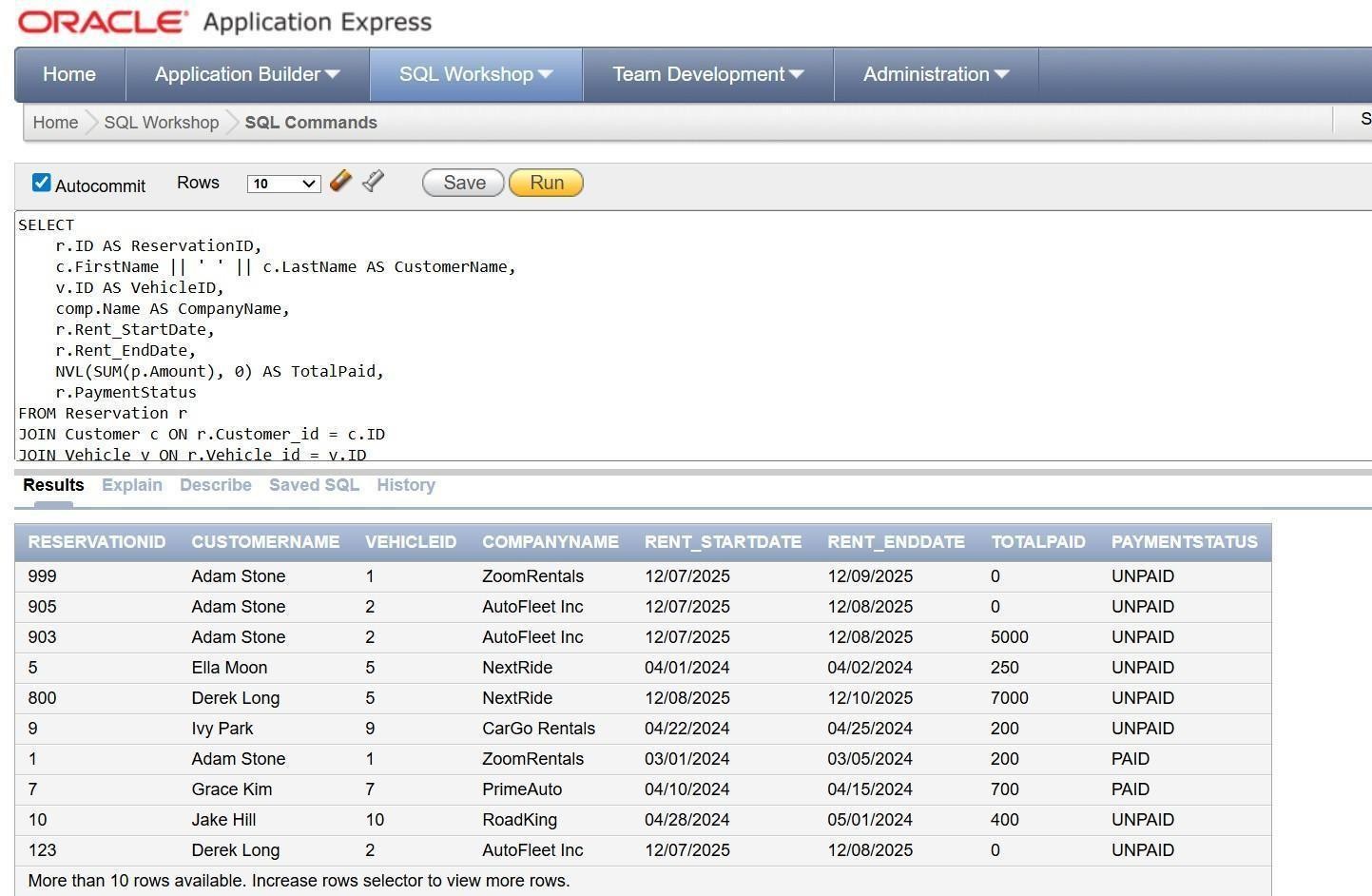
SELECT c.ID AS CustomerID, c.FirstName || ' ' || c.LastName AS CustomerName, c.Email, r.ID AS ReservationID, r.Status, r.PaymentStatus FROM Customer c LEFT JOIN Reservation r ON r.Customer\_id = c.ID WHERE c.BlackListStatus = 'Y' ORDER BY c.ID;



7. List All Active Reservations With Customer, Vehicle, Company & Payment Info.

SQL Query:

SELECT r.ID AS ReservationID, c.FirstName || ' ' || c.LastName AS CustomerName, v.ID AS VehicleID, comp.Name AS CompanyName, r.Rent\_StartDate, r.Rent\_EndDate, NVL(SUM(p.Amount), 0) AS TotalPaid, r.PaymentStatus FROM Reservation r JOIN Customer c ON r.Customer\_id = c.ID JOIN Vehicle v ON r.Vehicle\_id = v.ID JOIN Company comp ON v.Company\_id = comp.ID LEFT JOIN Payment p ON p.Reservation\_id = r.ID WHERE r.Status = 'Reserved' GROUP BY r.ID, c.FirstName, c.LastName, v.ID, comp.Name, r.Rent\_StartDate, r.Rent\_EndDate, r.PaymentStatus;



8.

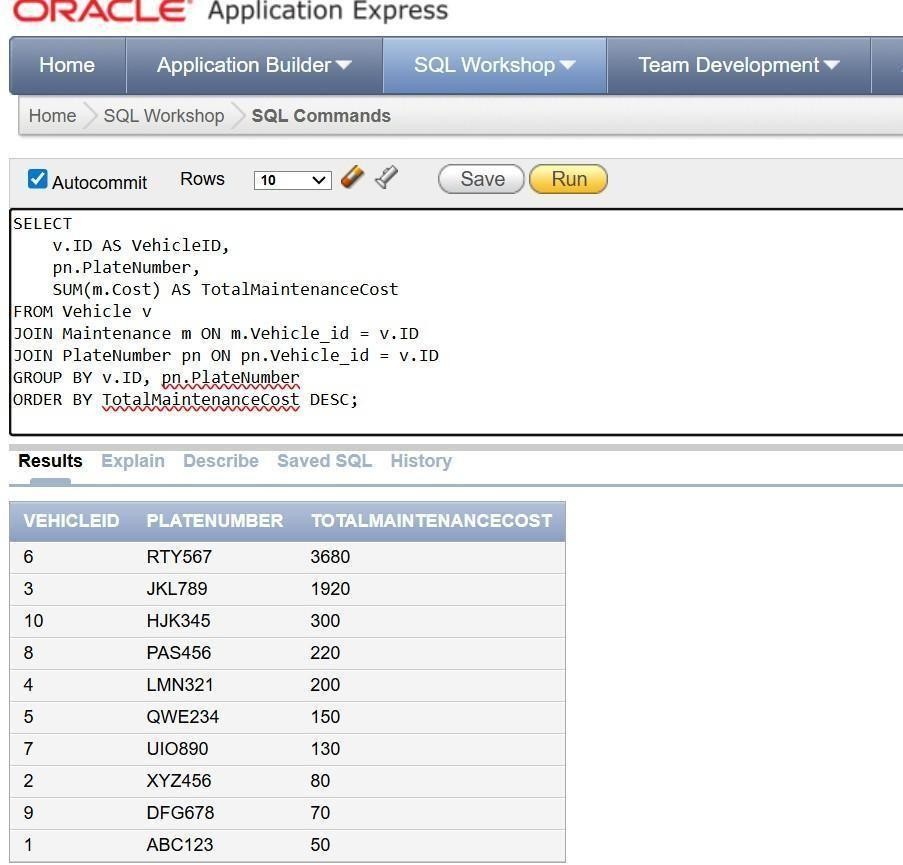
9.

10.

8.List Vehicles With Highest Maintenance Cost:

SQL Query

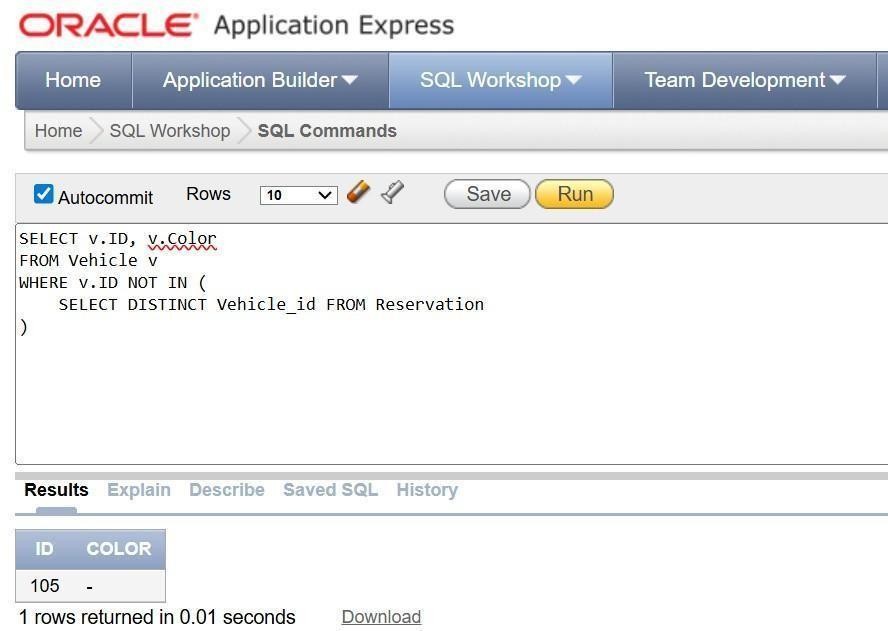
SELECT v.ID AS VehicleID, pn.PlateNumber, SUM(m.Cost) AS TotalMaintenanceCost FROM Vehicle v JOIN Maintenance m ON m.Vehicle\_id = v.ID JOIN PlateNumber pn ON pn.Vehicle\_id = v.ID GROUP BY v.ID, pn.PlateNumber ORDER BY TotalMaintenanceCost DESC;



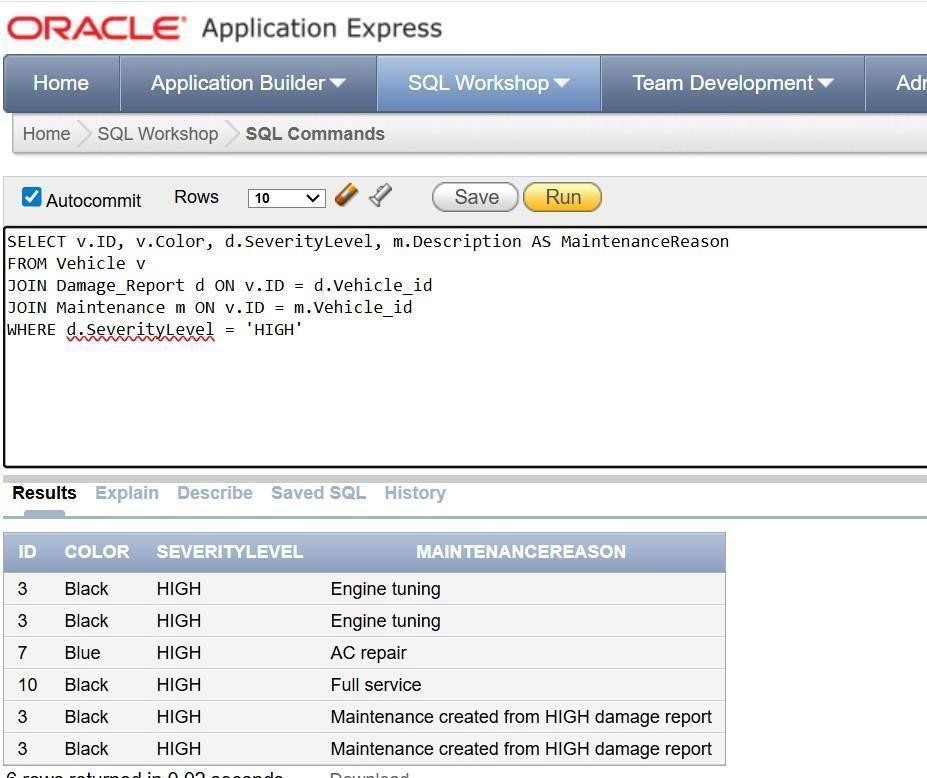
9. Which vehicles have never been reserved by any customer

SQL Query

SELECT v.ID, v.Color FROM Vehicle v WHERE v.ID NOT IN ( SELECT DISTINCT Vehicle\_id FROM Reservation )



9. Which vehicles required maintenance because of HIGH severity damage . SQL Query SELECT v.ID, v.Color, d.SeverityLevel, m.Description AS MaintenanceReason FROM Vehicle v JOIN Damage\_Report d ON v.ID = d.Vehicle\_id JOIN Maintenance m ON v.ID = m.Vehicle\_id WHERE d.SeverityLevel = 'HIGH'

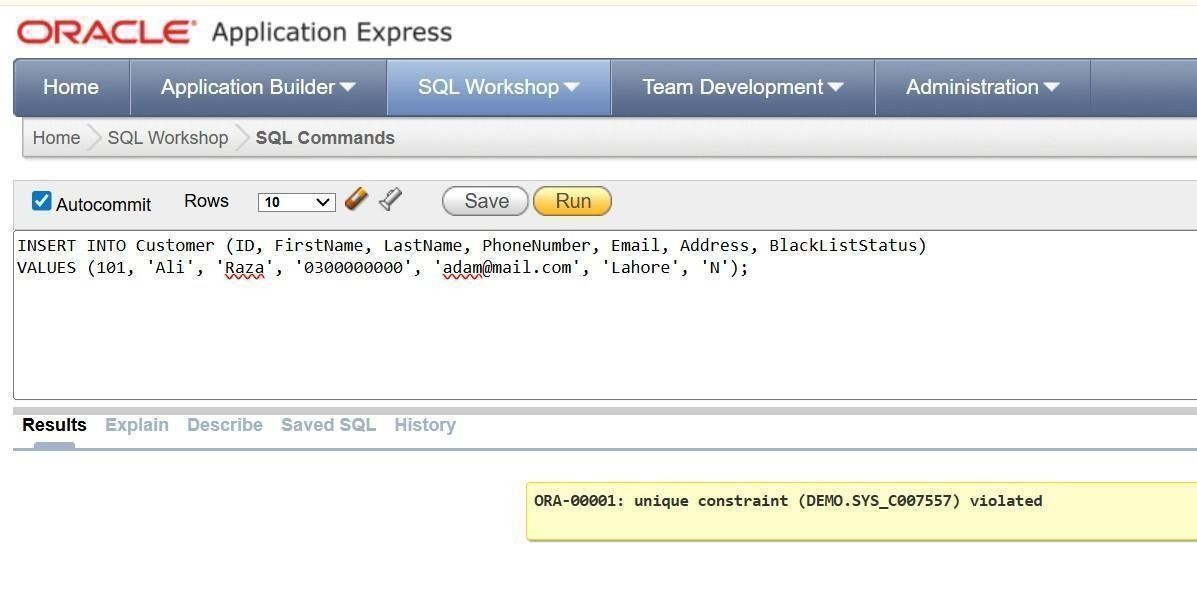


10.

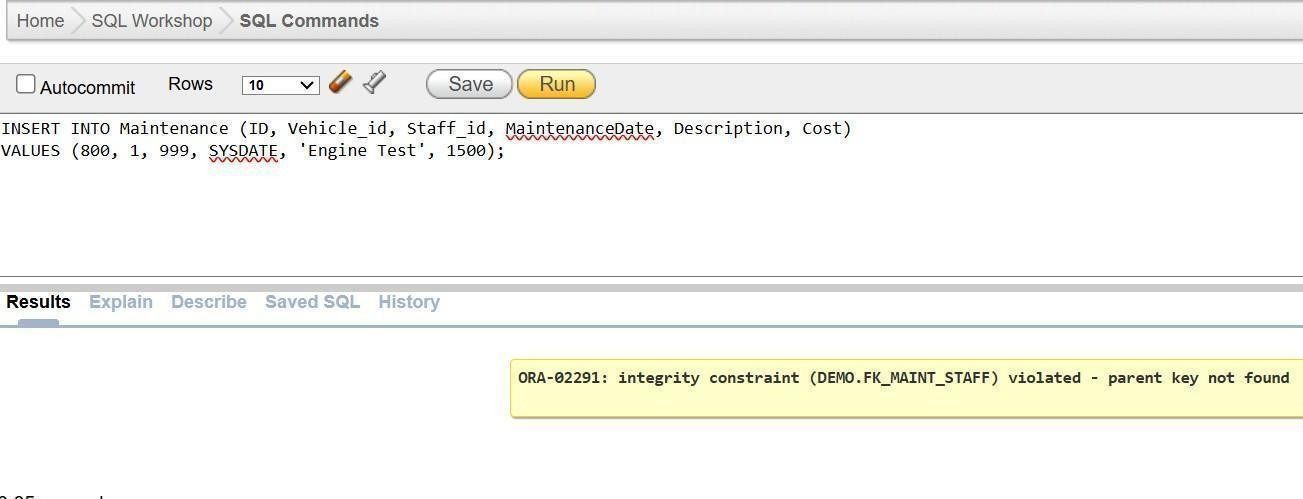
Testing And Error Handling: →UPDATE Reservation SET Status = 'Canceled' WHERE ID = 11; Expected: Test 1: Vehicle Status Returns to Active When Reservation is Canceled: →SELECT Status FROM Vehicle WHERE ID = 1; Vehicle status → Active



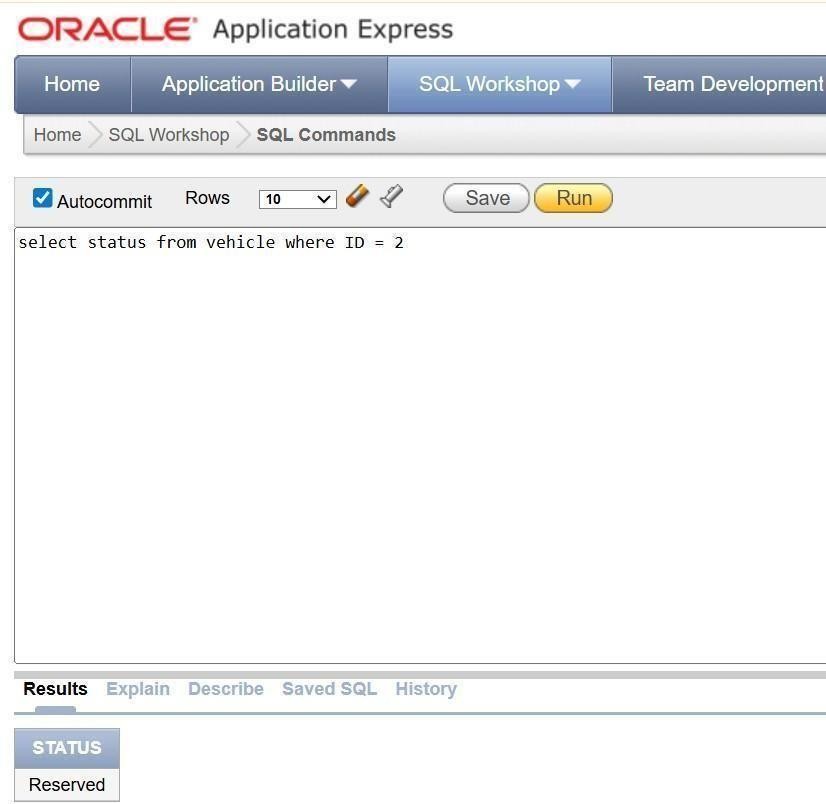
Test 2: Insert Customer With Duplicate Email: (Customer emails must be UNIQUE) The email ‘adam@mail.com’ already exists,so an error should be generated. →INSERT INTO Customer (ID, FirstName, LastName, PhoneNumber, Email, Address, BlackListStatus) VALUES (101, 'Ali', 'Raza', '0300000000', 'adam@mail.com', 'Lahore', 'N');



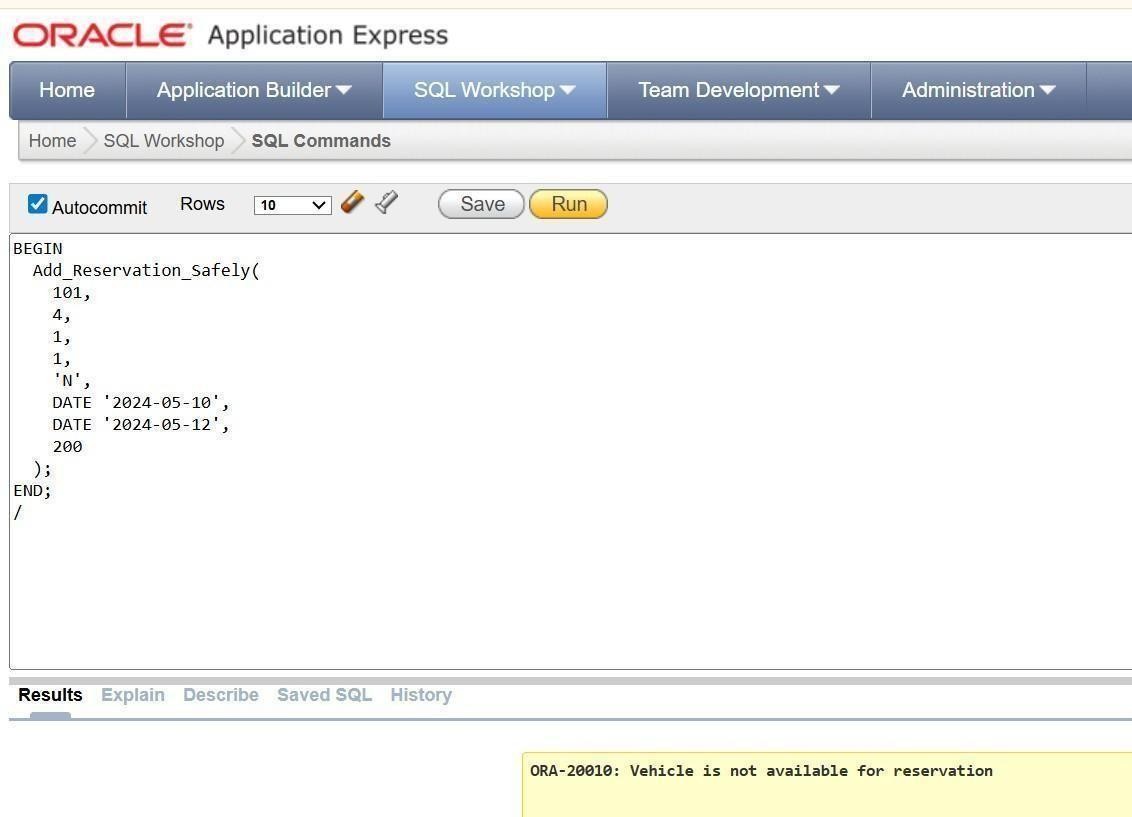
Test 3: Staff Id not existed in the Maintenance table (When we try to add maintenance data for a vehicle against a staff\_id who doesn’t exist) Expected: Integrity Constraint violation!!!!!



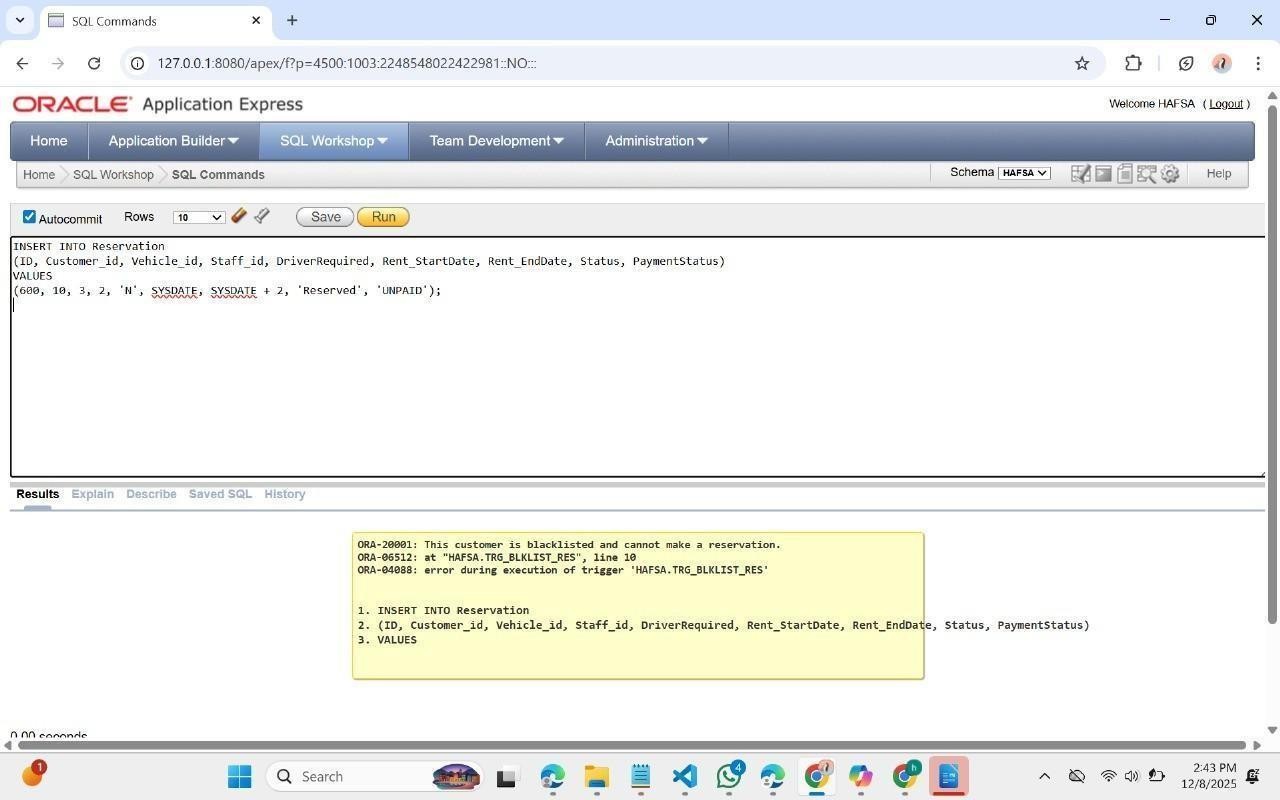
Test 4: Vehicle Status Automatically Changes to RESERVED INSERT Reservation → Vehicle.Status must become ‘Reserved’ \*INSERT INTO Reservation (ID, Customer\_id, Vehicle\_id, Staff\_id, DriverRequired, Rent\_StartDate, Rent\_EndDate, Status, PaymentStatus) VALUES (903, 1, 2, 1, 'N', SYSDATE, SYSDATE+1, 'Reserved', 'UNPAID'); \*SELECT Status FROM Vehicle WHERE ID = 2;

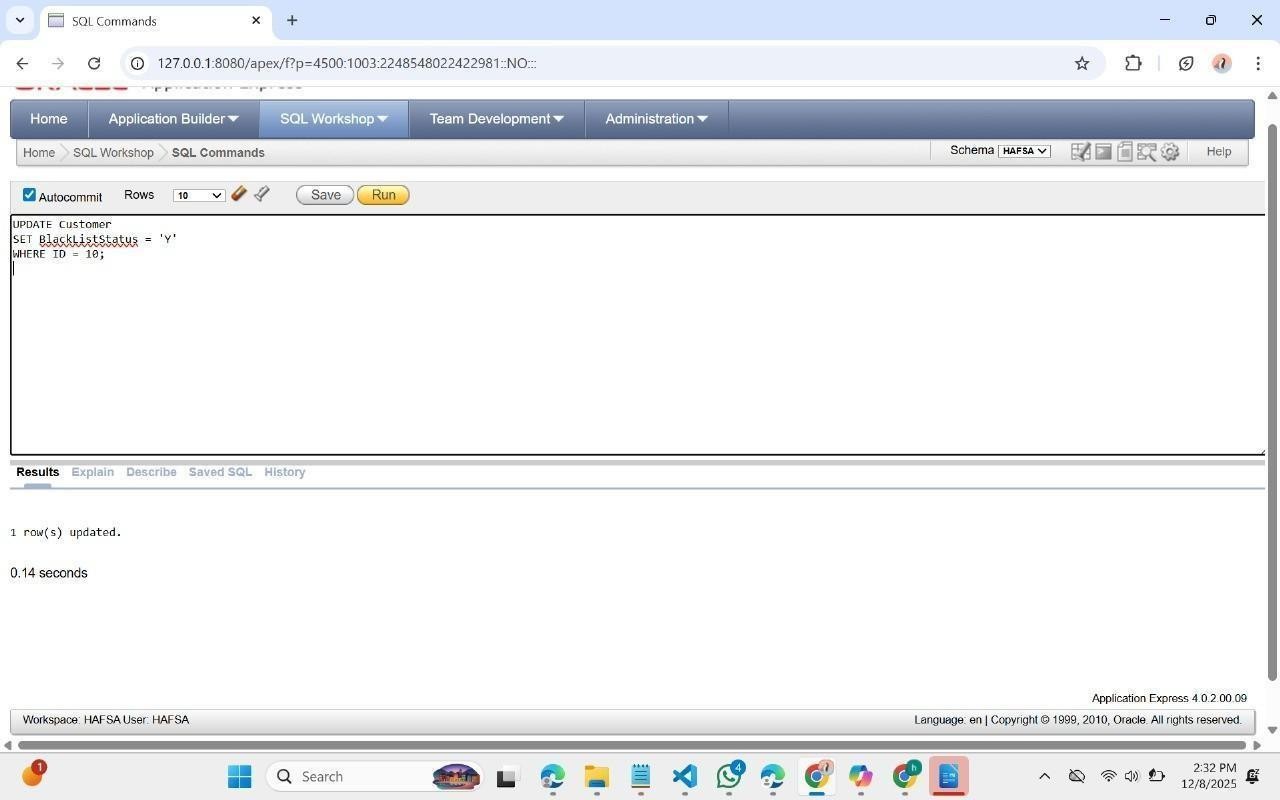


Test 5: Is the vehicle available or not: Check the availability of vehicle BEGIN Add\_Reservation\_Safely( ); END; / 101, 4, 1, 1, 'N', DATE '2024-05-10', DATE '2024-05-12', 200 Expected ERROR: ORA-20010: Vehicle is not available for reservation

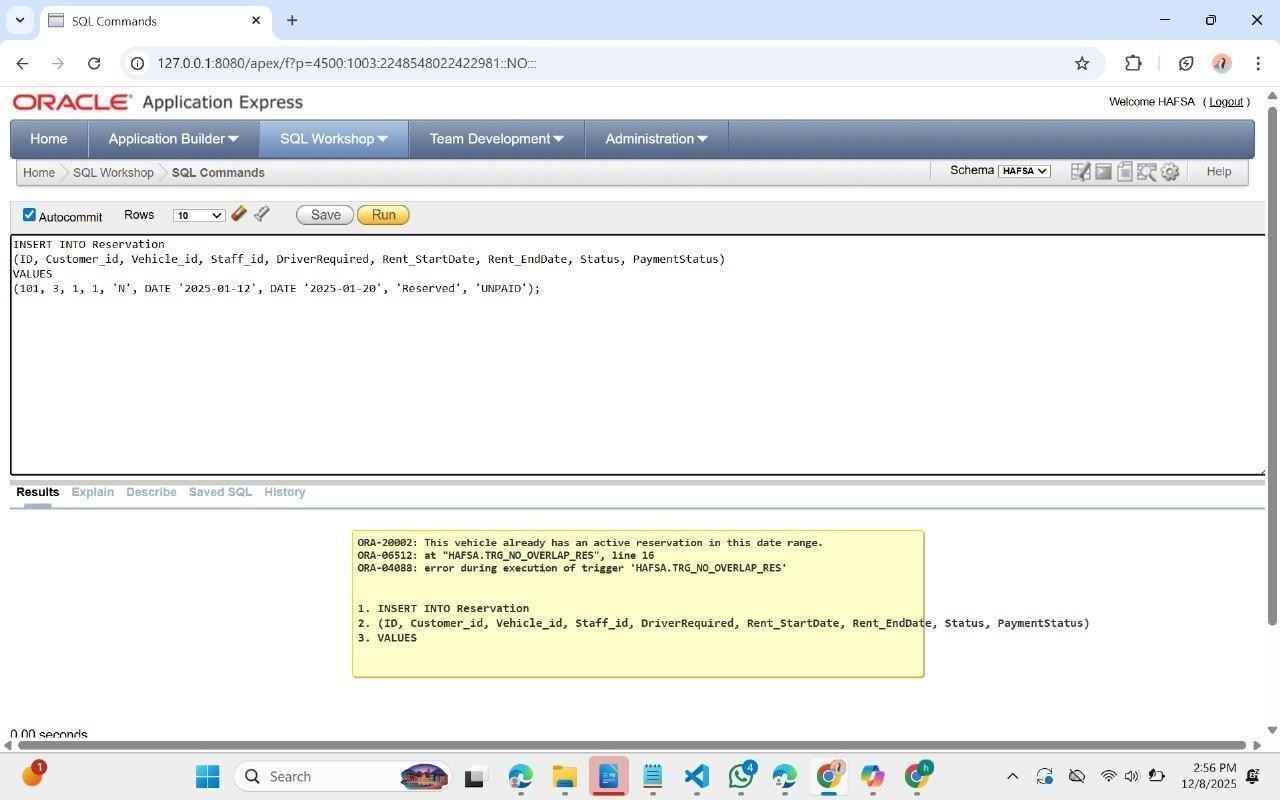


Test 6: Blacklist Customer cannot reserve a vehicle UPDATE Customer SET BlackListStatus = 'Y' WHERE ID = 10; (ID, Customer\_id, Vehicle\_id, Staff\_id, DriverRequired, Rent\_StartDate, Rent\_EndDate, Status, PaymentStatus) VALUES (501, 10, 3, 2, 'N', SYSDATE, SYSDATE + 2, 'Reserved', 'UNPAID'); Expected error: This customer is blacklisted and cannot make a reservation.

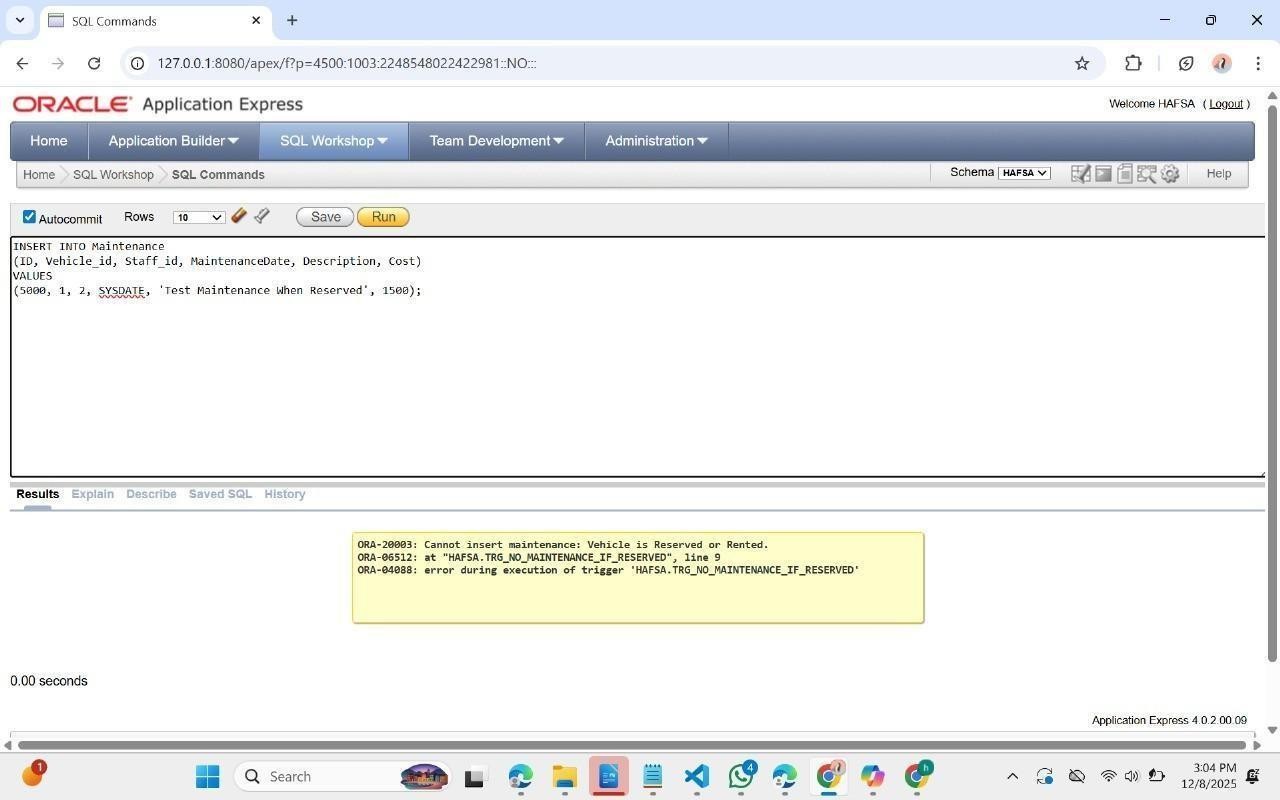




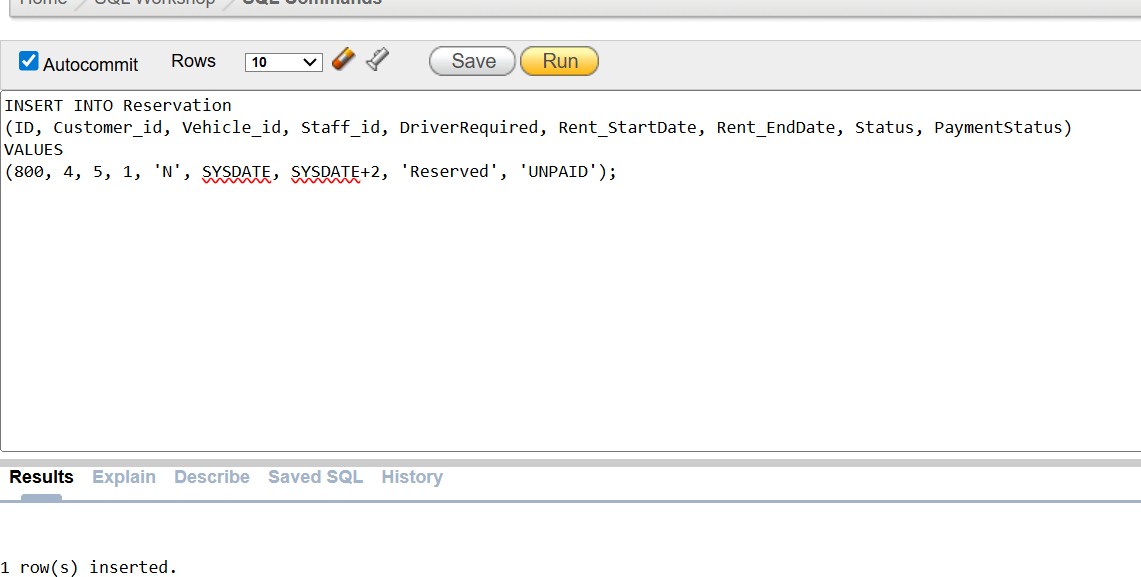
Test 7: No overlapping reservations (Reservations cannot overlap for the same vehicle) Expected Error : This vehicle already has an active reservation in this date range. INSERT INTO Reservation (ID, Customer\_id, Vehicle\_id, Staff\_id, DriverRequired, Rent\_StartDate, Rent\_EndDate, Status, PaymentStatus) VALUES (600, 2, 1, 1, 'N', DATE '2025-01-10', DATE '2025-01-15', 'Reserved', 'UNPAID'); ….. INSERT INTO Reservation (ID, Customer\_id, Vehicle\_id, Staff\_id, DriverRequired, Rent\_StartDate, Rent\_EndDate, Status, PaymentStatus) VALUES (601, 3, 1, 1, 'N', DATE '2025-01-12', DATE '2025-01-18', 'Reserved', 'UNPAID');



: Prevent Maintenance while vehicle is reserved UPDATE Vehicle SET Status = 'Reserved' WHERE ID = 3; INSERT INTO Maintenance (ID, Vehicle\_id, Staff\_id, MaintenanceDate, Description, Cost) VALUES (900, 3, 8, SYSDATE, 'Engine Tuning', 5000); Expected Error: Cannot insert maintenance: Vehicle is Reserved or Rented.



st 9: Only Mechanics Can Do Maintenance -- Suppose staff ID 5 is DRIVER role INSERT INTO Maintenance (ID, Vehicle\_id, Staff\_id, MaintenanceDate, Description, Cost) VALUES (901, 2, 5, SYSDATE, 'Oil Change', 2000); Expected error: Only mechanics (Other Staff role) can perform maintenance.



FINAL

PROFESSIONAL

TOUCH

Positive

Test

Cases

System

(

Should

Allow

These)

**Test**

**A:**

Successful

New

Reservation

INSERT INTO Reservation

(

ID,

Customer\_id,

Vehicle\_id,

Staff\_id,

DriverRequired,

Rent\_StartDate,

Rent\_EndDate,

Status, PaymentStatus)

VALUES

,

(800

4

,

5

,

1

,

'N',

SYSDATE,

SYSDATE+2,

'Reserved',

'UNPAID');

**Expected**

**Result:**

Reservation inserted successfully

Vehicle

5

→

Status

becomes

Reserved

