OYEEE TAXIII!!

CAB SERVICE

SUBMITTED BY :-
KRISHIV GOYAL - 102303864
BEDANGI SAHA - 102303867
NITYA GUPTA - 102303868
VEDIKA KAPOOR - 102313060
BATCH 2C62

SUBMITTED TO :-
SHIVANI GOSWAMI

INDEX

PROBLEM STATEMENT AND WORKING

- In today's world of busy city travel, cab companies need smart ways to keep their customers happy and loyal.
- Normal pricing often doesn't give any reward to regular users.
- Our project solves this by creating a smart discount system in a cab booking platform. It gives discounts based on total fares spent by a customer and how often they book rides.
- It also adds loyalty points for each ride, which can be used for future discounts.
- A carpool feature is added, so customers can share rides and save money.
- The backend uses PL/SQL code to automatically apply discounts, calculate fares, and handle loyalty points. This makes the service more user-friendly, fair, and better for the environment.

NOVELTY

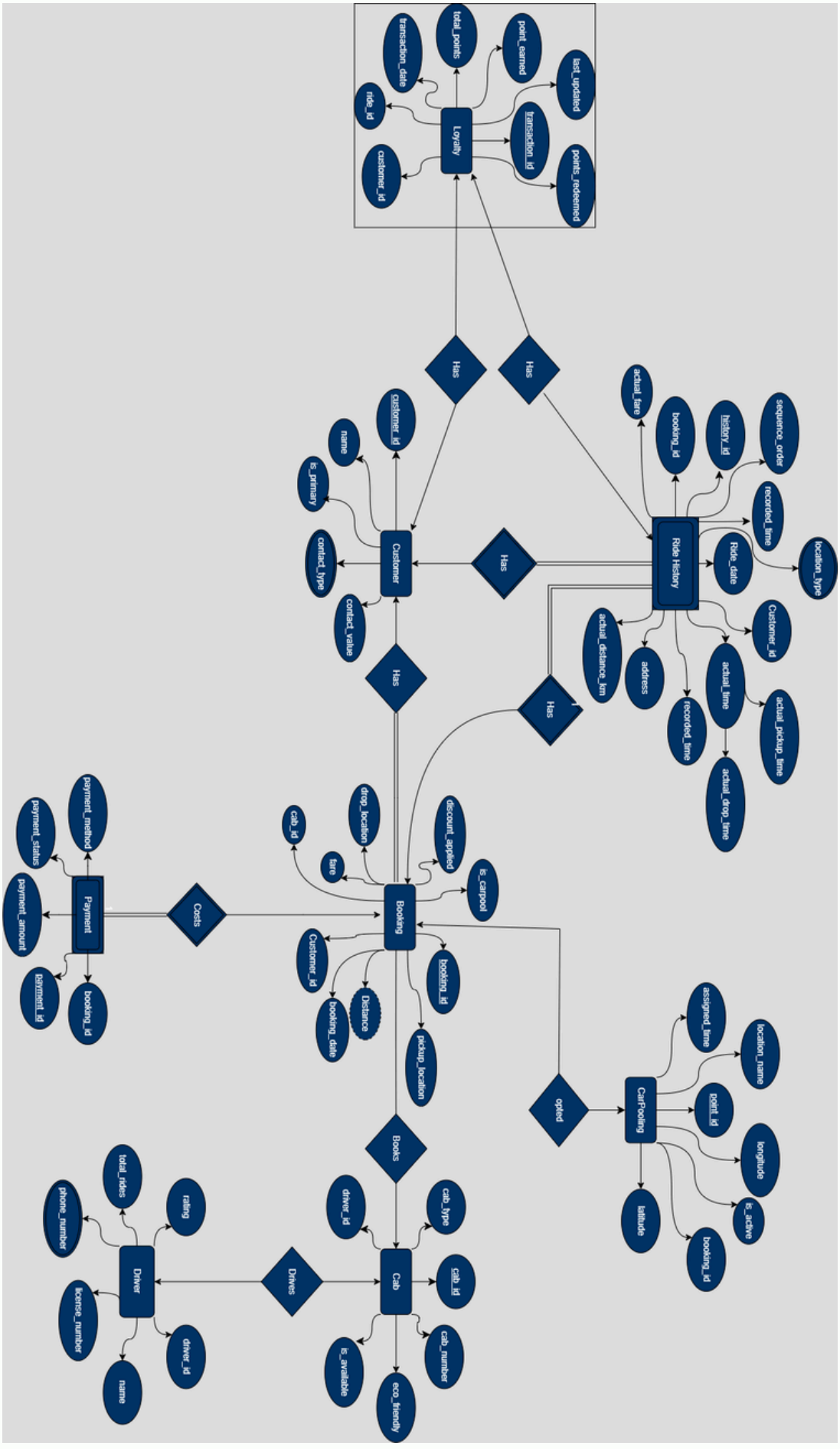
- Loyalty Point System: Unlike traditional cab services, our system introduces a loyalty points feature where users earn points for ride. These points can be redeemed later for discounts, encouraging frequent usage and increasing customer satisfaction.

- Carpooling Feature: The system includes a built-in carpooling option that allows multiple customers to share a single ride. This not only reduces the cost for passengers but also helps reduce traffic and pollution, promoting an eco-friendly transport solution.

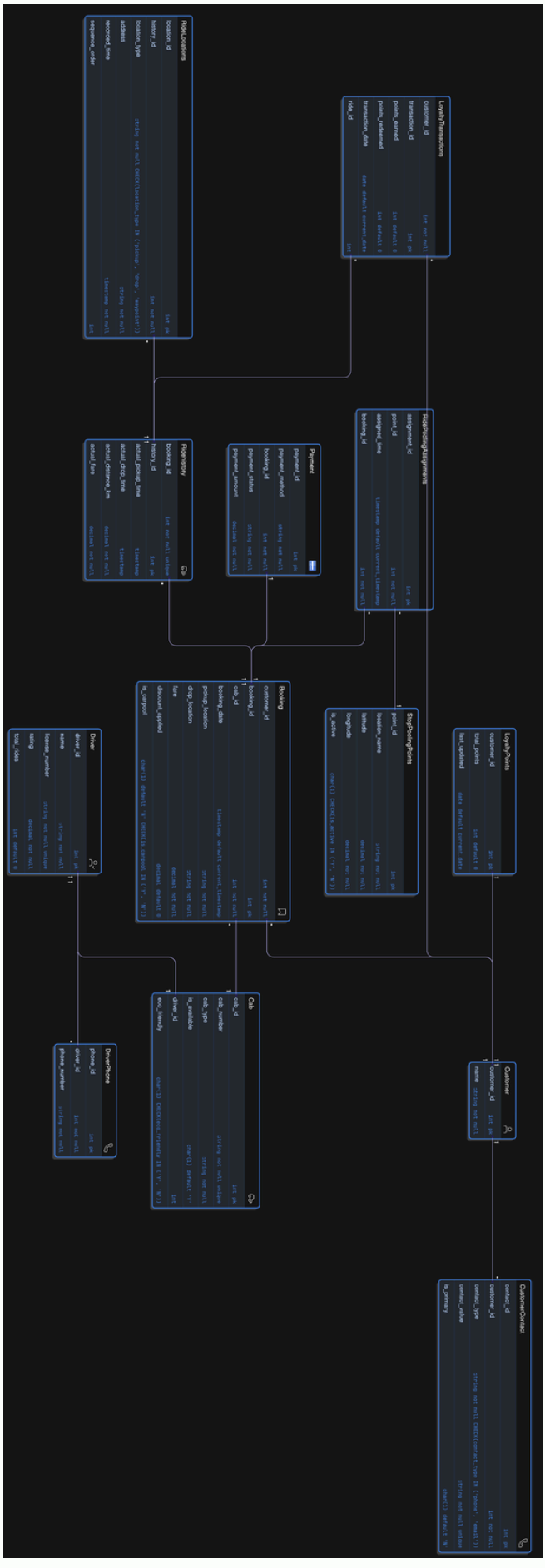
- Integrated Discount Logic: The discount system is dynamic and personalized—it considers the number of bookings to apply fair discounts. This makes the pricing model more engaging and beneficial for regular users.

These features set our project apart from standard cab booking systems and make it more practical, rewarding, and sustainable for both customers and service providers.

ER DIAGRAM


Loyalty
total_points
point_earned
last_updated
transaction_date
transaction_id
ride_id
customer_id
points_redeemed
Has
Has
Customer
customer_id
name
is_primary
contact_type
contact_value
Has
Ride History
actual_fare
booking_id
history_id
sequence_order
recorded_time
Ride_date
location_type
Customer_id
actual_time
actual_pickup_time
actual_drop_time
recorded_time
address
actual_distance_km
Has
Has
Booking
cab_id
drop_location
fare
discount_applied
is_carpool
Customer_id
booking_date
Distance
booking_id
pickup_location
Payment
payment_method
payment_status
payment_amount
payment_id
booking_id
Costs
Books
Drives
Cab
driver_id
cab_type
cab_id
is_available
cab_number
eco_friendly
Driver
total_rides
rating
phone_number
license_number
driver_id
name
opted
CarPooling
assigned_time
location_name
point_id
longitude
is_active
latitude
booking_id

RELATIONAL TABLES



NORMALIZED TABLES

1. Customer
   - customer_id: INT PRIMARY KEY
   - name: VARCHAR2(100) NOT NULL
2. CustomerContact
   - contact_id: NUMBER PRIMARY KEY
   - customer_id: NUMBER NOT NULL
   - → FOREIGN KEY to Customer(customer_id) ON DELETE CASCADE
   - contact_type: VARCHAR2(10) NOT NULL
   - → CHECK (IN ('phone', 'email'))
   - contact_value: VARCHAR2(100) NOT NULL
   - → UNIQUE
   - is_primary: CHAR(1) DEFAULT 'N'
   - Constraints:
     - UNIQUE(customer_id, contact_type, is_primary)
     - → Only one primary contact per type per customer
3. Driver
   - driver_id: NUMBER PRIMARY KEY
   - name: VARCHAR2(100) NOT NULL
   - license_number: VARCHAR2(50) NOT NULL UNIQUE
   - rating: DECIMAL(3, 2) NOT NULL
   - total_rides: NUMBER DEFAULT 0
4. DriverPhone
   - phone_id: NUMBER PRIMARY KEY
   - driver_id: NUMBER NOT NULL
   - → FOREIGN KEY to Driver(driver_id) ON DELETE CASCADE
   - phone_number: VARCHAR2(15) NOT NULL
   - Constraint: UNIQUE(driver_id, phone_number)
5. Cab
   - cab_id: NUMBER PRIMARY KEY
   - cab_number: VARCHAR2(20) NOT NULL UNIQUE
   - cab_type: VARCHAR2(50) NOT NULL
   - is_available: CHAR(1) DEFAULT 'Y'
   - driver_id: NUMBER
   - → FOREIGN KEY to Driver(driver_id) ON DELETE SET NULL
   - eco_friendly: CHAR(1)
   - → CHECK (IN ('Y', 'N'))

NORMALIZED TABLES

6. Booking
   - booking_id: NUMBER PRIMARY KEY
   - customer_id: NUMBER NOT NULL
   - → FOREIGN KEY to Customer(customer_id)
   - cab_id: NUMBER NOT NULL
   - → FOREIGN KEY to Cab(cab_id)
   - booking_date: TIMESTAMP DEFAULT CURRENT_TIMESTAMP
   - pickup_location: VARCHAR2(255) NOT NULL
   - drop_location: VARCHAR2(255) NOT NULL
   - fare: DECIMAL(10, 2) NOT NULL
   - discount_applied: DECIMAL(10, 2) DEFAULT 0
   - is_carpool: CHAR(1) DEFAULT 'N'
   - → CHECK (IN ('Y', 'N'))
7. RideHistory
   - history_id: NUMBER PRIMARY KEY
   - booking_id: NUMBER NOT NULL UNIQUE
   - → FOREIGN KEY to Booking(booking_id)
   - actual_pickup_time: TIMESTAMP
   - actual_drop_time: TIMESTAMP
   - actual_distance_km: NUMBER(6, 2) NOT NULL
   - actual_fare: NUMBER(10, 2) NOT NULL
8. RideLocations
   - location_id: NUMBER PRIMARY KEY
   - history_id: NUMBER NOT NULL
   - → FOREIGN KEY to RideHistory(history_id) ON DELETE CASCADE
   - location_type: VARCHAR2(10) NOT NULL
   - → CHECK (IN ('pickup', 'drop', 'waypoint'))
   - address: VARCHAR2(255) NOT NULL
   - recorded_time: TIMESTAMP NOT NULL
   - sequence_order: NUMBER
   - Constraint: UNIQUE(history_id, sequence_order)
9. Payment
   - payment_id: NUMBER PRIMARY KEY
   - booking_id: NUMBER NOT NULL
   - → FOREIGN KEY to Booking(booking_id) ON DELETE CASCADE
   - payment_method: VARCHAR2(50) NOT NULL
   - payment_status: VARCHAR2(50) NOT NULL
   - payment_amount: NUMBER(6, 2) NOT NULL

NORMALIZED TABLES

10. StopPoolingPoints
    - point_id: NUMBER PRIMARY KEY
    - location_name: VARCHAR2(100) NOT NULL
    - latitude: NUMBER(9, 6) NOT NULL
    - longitude: NUMBER(9, 6) NOT NULL
    - is_active: CHAR(1)
    - → CHECK (IN ('Y', 'N'))
11. RidePoolingAssignments
    - assignment_id: NUMBER PRIMARY KEY
    - booking_id: NUMBER NOT NULL
    - → FOREIGN KEY to Booking(booking_id) ON DELETE CASCADE
    - point_id: NUMBER NOT NULL
    - → FOREIGN KEY to StopPoolingPoints(point_id)
    - assigned_time: TIMESTAMP DEFAULT SYSTIMESTAMP
12. LoyaltyPoints
    - customer_id: NUMBER PRIMARY KEY
    - → FOREIGN KEY to Customer(customer_id)
    - total_points: NUMBER DEFAULT 0
    - last_updated: DATE DEFAULT SYSDATE
13. LoyaltyTransactions
    - transaction_id: NUMBER PRIMARY KEY
    - customer_id: NUMBER NOT NULL
    - → FOREIGN KEY to Customer(customer_id)
    - ride_id: NUMBER
    - → FOREIGN KEY to RideHistory(history_id)
    - points_earned: NUMBER DEFAULT 0
    - points_redeemed: NUMBER DEFAULT 0
    - transaction_date: DATE DEFAULT SYSDATE

PL/SQL

```
Customer        Booking ID  Orig Fare   Final Fare  Discount %  Loyalty Used   Remain LP   Reason
-------------------------------------------------------------------------------------------------------------------
Amit Sharma     1           ₹ 200.00    ₹ 153.00    25%         500            0           Loyalty discount + carpool
Amit Sharma     2           ₹ 250.00    ₹ 225.00    10%         0              0           Loyalty discount + carpool
Amit Sharma     3           ₹ 180.00    ₹ 145.80    20%         0              0           Booking-based discount + carpool
Neha Verma      4           ₹ 220.00    ₹ 168.30    25%         500            100         Loyalty discount + carpool
Rahul Gupta     5           ₹ 210.00    ₹ 160.65    25%         500            200         Loyalty discount + carpool
Priya Singh     10          ₹ 280.00    ₹ 238.00    15%         500            300         Loyalty discount
Priya Singh     7           ₹ 240.00    ₹ 216.00    10%         200            100         Loyalty discount
Priya Singh     8           ₹ 260.00    ₹ 234.00    10%         0              100         Booking-based discount
Priya Singh     9           ₹ 270.00    ₹ 229.50    15%         0              100         Booking-based discount
Priya Singh     6           ₹ 230.00    ₹ 184.00    20%         0              100         Booking-based discount

✅ Discounts applied and database updated successfully.
```

```
DECLARE
  CURSOR booking_cursor IS
    SELECT b.booking_id, b.customer_id, b.cab_id, b.fare, b.is_carpool, b.booking_date,
        c.name AS customer_name, d.name AS driver_name, cb.cab_number, cb.cab_type
    FROM Booking b
    JOIN Customer c ON b.customer_id = c.customer_id
    JOIN Cab cb ON b.cab_id = cb.cab_id
    JOIN Driver d ON cb.driver_id = d.driver_id
    ORDER BY b.customer_id, b.booking_date;
  v_booking_sequence NUMBER := 0;
  v_prev_customer_id NUMBER := 0;
  v_current_quarter_start DATE;
  v_booking_distance_discount NUMBER := 0;
  v_loyalty_discount NUMBER := 0;
  v_final_discount NUMBER := 0;
  v_final_fare NUMBER;
  v_carpool_discount NUMBER := 0;
  v_discount_reason VARCHAR2(200);
  v_loyalty_points NUMBER;
  v_loyalty_points_used NUMBER := 0;
  v_first_output_done BOOLEAN := FALSE;
BEGIN
  FOR booking_rec IN booking_cursor LOOP
    IF v_prev_customer_id != booking_rec.customer_id OR
      TRUNC(booking_rec.booking_date, 'Q') != v_current_quarter_start THEN
      v_booking_sequence := 0;
      v_current_quarter_start := TRUNC(booking_rec.booking_date, 'Q');
      v_prev_customer_id := booking_rec.customer_id;
    END IF;
    v_booking_sequence := v_booking_sequence + 1;
    v_booking_distance_discount := 0;
    v_loyalty_discount := 0;
    v_final_discount := 0;
    v_discount_reason := '';
    v_carpool_discount := 0;
    v_loyalty_points_used := 0;
```

PL/SQL

```
-- Booking sequence discount
IF v_booking_sequence = 3 THEN
v_booking_distance_discount := 10;
v_discount_reason := '3rd booking in quarter';
ELSIF v_booking_sequence = 4 THEN
v_booking_distance_discount := 15;
v_discount_reason := '4th booking in quarter';
ELSIF v_booking_sequence >= 5 THEN
v_booking_distance_discount := 15 + (5 * (v_booking_sequence - 4));
IF v_booking_distance_discount > 50 THEN
v_booking_distance_discount := 50;
END IF;
v_discount_reason := v_booking_sequence || 'th booking in quarter';
END IF;

-- Loyalty discount
SELECT total_points INTO v_loyalty_points
FROM LoyaltyPoints
WHERE customer_id = booking_rec.customer_id;

IF v_loyalty_points >= 1000 THEN
v_loyalty_discount := 20;
v_loyalty_points_used := 1000;
ELSIF v_loyalty_points >= 500 THEN
v_loyalty_discount := 15;
v_loyalty_points_used := 500;
ELSIF v_loyalty_points >= 200 THEN
v_loyalty_discount := 10;
v_loyalty_points_used := 200;
ELSIF v_loyalty_points >= 100 THEN
v_loyalty_discount := 5;
v_loyalty_points_used := 100;
END IF;

-- Final discount decision
IF v_booking_distance_discount > v_loyalty_discount THEN
v_final_discount := v_booking_distance_discount;
v_discount_reason := 'Booking-based discount';
v_loyalty_points_used := 0;
ELSE
v_final_discount := v_loyalty_discount;
v_discount_reason := 'Loyalty discount';
END IF;
```

PL/SQL

```
-- Carpool discount
IF booking_rec.is_carpool = 'Y' THEN
v_carpool_discount := 10;
v_discount_reason := v_discount_reason || ' + carpool';
END IF;

-- Final fare calculation
v_final_fare := booking_rec.fare * (1 - v_final_discount/100) * (1 - v_carpool_discount/100);

-- Update Booking table
UPDATE Booking
SET fare = v_final_fare,
discount_applied = v_final_discount + v_carpool_discount
WHERE booking_id = booking_rec.booking_id;

-- Deduct loyalty points
IF v_loyalty_points_used > 0 THEN
UPDATE LoyaltyPoints
SET total_points = total_points - v_loyalty_points_used
WHERE customer_id = booking_rec.customer_id;

UPDATE LoyaltyTransactions
SET points_redeemed = v_loyalty_points_used
WHERE customer_id = booking_rec.customer_id;
END IF;

-- Print formatted table header once
IF NOT v_first_output_done THEN
DBMS_OUTPUT.PUT_LINE(
RPAD('Customer', 20) || RPAD('Booking ID', 12) || RPAD('Orig Fare', 12) ||
RPAD('Final Fare', 12) || RPAD('Discount %', 12) || RPAD('Loyalty Used', 15) ||
RPAD('Remain LP', 12) || 'Reason'
);
DBMS_OUTPUT.PUT_LINE(RPAD('-', 120, '-'));
v_first_output_done := TRUE;
END IF;
```

PL/SQL

```
-- Print row output
DBMS_OUTPUT.PUT_LINE(
RPAD(booking_rec.customer_name, 20) ||
RPAD(booking_rec.booking_id, 12) ||
RPAD('₹' || TO_CHAR(booking_rec.fare, '990.00'), 12) ||
RPAD('₹' || TO_CHAR(v_final_fare, '990.00'), 12) ||
RPAD(TO_CHAR(v_final_discount + v_carpool_discount) || '%', 12) ||
RPAD(v_loyalty_points_used, 15) ||
RPAD(v_loyalty_points - v_loyalty_points_used, 12) ||
v_discount_reason
);
END LOOP;

COMMIT;
DBMS_OUTPUT.PUT_LINE(CHR(10) || '✅ Discounts applied and database updated successfully.');
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('❌ Error occurred: ' || SQLERRM);
ROLLBACK;
END;
/
```

ROLES AND CONTRIBUTIONS

Krishiv Goyal - PL/SQL(Discount and Carpool), Frontend design, ER diagram
Bedangi Saha - Frontend, SQL Query, Login Authentication
Nitya Gupta - Frontend, Login Backend, Novelty Idea, PL/SQL(Loyalty Points), API
Vedika - Linking Database, Backend, Frontend, SQL Query
Team - Database Creation, Bug fixing, Normaliztion

CONCLUSION

- The project successfully delivers a smart cab discount and loyalty system tailored for modern users.
- It tracks customer rides, distances, amount spent, and booking frequency to calculate personalized discounts.
- Loyalty points are awarded for rides, encouraging repeat usage and long-term customer relations.
- PL/SQL procedures, functions, and cursor ensure backend processing and real-time updates.
- The carpooling feature promotes cost savings and supports eco-friendly travel.
- Overall, the system increases user satisfaction.

REFERENCE

1. Oracle Library
2. W3Schools SQL Tutorial
3. OpenAI for query enter and query bug fixing

SIGNATURE