

# TripSync: Project Deadline-5

## Triggers:

### Trigger 1:

```
CREATE TRIGGER lock_account_trigger BEFORE UPDATE ON userids_passwords
FOR EACH ROW
BEGIN
    IF NEW.tries >= 3 THEN
        SET NEW.is_locked = 'T';
        SET NEW.attempt_time = CURRENT_TIMESTAMP;
    END IF;
END //

-- DELIMITER ;

-- Dropping the unlock_account_event if it exists
DROP EVENT IF EXISTS unlock_account_event;

-- Recreating the unlock_account_event event only if it doesn't already exist
CREATE EVENT IF NOT EXISTS unlock_account_event
ON SCHEDULE EVERY 10 SECOND
STARTS CURRENT_TIMESTAMP
DO
    UPDATE userids_passwords
    SET is_locked = 'F', tries = 0
    WHERE is_locked = 'T';

-- DELIMITER ;

-- Dropping the update_valid_tickets_trigger if it exists
DROP TRIGGER IF EXISTS update_valid_tickets_trigger;

-- Recreating the update_valid_tickets_trigger trigger
DELIMITER //
```

Explanation: The above trigger works such that if a user enters the password for their user id wrong **more than three times** then this trigger would automatically lock the user's account for 10 seconds after which it will be unlocked automatically by the event "unlock\_account\_event". This event protects the user from losing access to his/her account forever.

This trigger is made to protect the user's account from being accessed by a hacker who might try to gain access by putting in multiple passwords one after another.

## Trigger 2:

```
CREATE TRIGGER update_valid_tickets_trigger BEFORE INSERT ON Tickets
FOR EACH ROW
BEGIN
    IF NEW.date_of_journey < NOW() THEN
        SET NEW.is_valid = 0;
    #      insert into tickets (Ticket_No, Train_No, Flight_No, Amount, Date_of_journey, Quantity, userid) VALUES (NEW.Ticket_No,NEW.Train_No,NEW.Flight_No,NEW.Amount,NEW.Date_of_journey,NE
    END IF;
    #      insert into tickets (Ticket_No, Train_No, Flight_No, Amount, Date_of_journey, Quantity, userid) VALUES (NEW.Ticket_No,NEW.Train_No,NEW.Flight_No,NEW.Amount,NEW.Date_of_journey,NE
    END//

-- DELIMITER ;

DELIMITER //

-- Dropping the validate_tickets_event if it exists
DROP EVENT IF EXISTS validate_tickets_event;

# -- Recreating the validate_tickets_event event only if it doesn't already exist
CREATE EVENT IF NOT EXISTS validate_tickets_event
ON SCHEDULE EVERY 10 SECOND
STARTS CURRENT_TIMESTAMP
DO
    UPDATE Tickets
    SET is_valid = 0
    WHERE date_of_journey < NOW();

# DELIMITER //
```

## Explanation:

This trigger will help maintain validity and data integrity in the **Tickets** table .

Suppose a user tries to book a ticket for 31 March , 2024 but the current date is already 1st April , 2024 . Without triggering the is\_valid column corresponding to the relevant ticket to 0 in the Ticket table , there would be no way of intercepting this anomaly . So, it makes sense to incorporate a check which compares the date\_of\_journey to the present date and sets the ticket.is\_valid to 0 if the ticket has already expired before inserting it into the database .

The event validate\_tickets\_event essentially serves as a periodic check which happens every 10 seconds to invalidate any ticket records that have a journey date in the past . It complements the trigger by ensuring that any records missed by the trigger are also marked as invalid.

## Trigger 3:

```

DROP TRIGGER IF EXISTS update_valid_tickets_trigger7;

create trigger update_valid_tickets_trigger7 BEFORE update on tickets
FOR EACH ROW
BEGIN
    IF NEW.date_of_journey > NOW() THEN
        SET NEW.is_valid = 1;
    ELSE
        SET NEW.is_valid = 0;
    END IF;
END;

```

Explanation: The above trigger is used to validate the ticket in case the 2nd trigger works.

Suppose in case there was a train at 8:00 pm and at 8 pm, the train doesn't arrive, in that case, due to the 2nd trigger, the valid bit will be set 0, but it shouldn't be, so we set the valid bit to 1 again which will signify that the train ticket is still valid.

This trigger will only make the changes in the Tickets table is\_valid column entries.

## **Embedded SQL Queries:**

```

import random
import time
# # Create a new user
# user = User.objects.create_user(username='example_user',
password='example_password')
from datetime import datetime

import mysql.connector

```

**(Here I have imported necessary packages to connect )**

```
def mysql_bckens():
    connection = mysql.connector.connect(
        host='localhost', # host mt hi chedo chl jayega
        user='root', # user dalo
        password='Aditya@1998', # password dalo me bdl dung anhi batunga
        database='makemytrip' # optional hai comment bhi kr skte ho phir
        password me se , bhi remove krna
    )
    if connection.is_connected():
        print("YES")
    else:
        print("NO"*100)

    # Create cursor
    # cursor = connection.cursor()

    return connection
```

**(This is the function where user enters his/her user and password it is important to make connection)**

## **Query 1**

```
adi_conn1 = mysql_bckens()
adi_conn = adi_conn1.cursor()

# Query 1 if we want to see or all female users

adi_conn.execute("Select count(*) from users where Gender = (%s) group by
Gender",('F',))

k = adi_conn.fetchall()

for i in k:
    print("Total female users are", i[0])
```

**(IF we want to see all female customers/users)**

## **Query 2**

```
# Quer 2 if we want to see which of our current users have ever booked any hotel
```

```
adi_conn.execute("SELECT name FROM users WHERE userid IN (SELECT DISTINCT  
userid FROM hotel_invoice)")
```

```
k = adi_conn.fetchall()  
for i in k:  
    print(i[0])
```

**(# Query 2 if we want to see which of our current users have ever booked any hotel**  
**)**

### **Query 3**

```
# Query 3 Here i printed those unique users whose payment has succes that  
means they have booked from our application
```

```
adi_conn.execute("Select Distinct user_id from payments where payment_status =  
(%s)",(1,))
```

```
k = adi_conn.fetchall()  
for i in k:  
    print(i[0])
```

**(Query 3 Here i printed those unique users whose payment has succes that means they have booked from our application**  
**)**

## **How to run the code:**

1. To clone just run  
*git clone https://github.com/op2adi/DBMS\_Main\_project.git*
2. Change the directory to Tripsync after cloning the repository.
3. Run the following command on your terminal. “ python ./manage.py  
runserver “
4. Then you simply need to run on any search engine <http://127.0.0.1:8000/>  
(or simply click on it)
5. Bingo Here you go; You landed on login page.