CREATE DEFINER=`root`@`localhost` PROCEDURE `book\_ticket`(

IN p\_email VARCHAR(50),

IN p\_train\_id VARCHAR(6),

IN p\_class\_name VARCHAR(2),

IN p\_date DATE

)

BEGIN

DECLARE seatno INT;

DECLARE ticketno INT;

DECLARE fare\_ INT;

SELECT get\_max\_seat\_number(p\_train\_id, p\_class\_name, p\_date) INTO seatno;

IF seatno IS NULL THEN

SET seatno = 1;

ELSE

SET seatno = seatno + 1;

END IF;

INSERT INTO confirmed\_seats (train\_id, class\_name, seat\_no, date\_of\_journey)

VALUES (p\_train\_id, p\_class\_name, seatno, p\_date);

SELECT COUNT(\*) + 1 INTO ticketno

FROM bookings;

SELECT fare INTO fare\_

FROM seats

WHERE train\_id = p\_train\_id AND class\_name = p\_class\_name;

IF fare\_ IS NULL THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Fare not found for the specified train and class';

END IF;

INSERT INTO bookings

VALUES (ticketno, p\_train\_id, p\_class\_name, seatno,

(SELECT origin FROM train\_tt WHERE train\_id = p\_train\_id AND date\_of\_journey = DATE(p\_date)),

(SELECT destination FROM train\_tt WHERE train\_id = p\_train\_id AND date\_of\_journey = DATE(p\_date)),

p\_date, CURDATE(), p\_email,

fare\_, 'B');

UPDATE user\_profile

SET money = money - fare\_

WHERE email = p\_email;

END

2.

CREATE DEFINER=`root`@`localhost` PROCEDURE `fetch\_passengers`(

IN p\_train\_id VARCHAR(6),

IN p\_date\_of\_journey DATE

)

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE ticketno INT;

DECLARE c1 CURSOR FOR SELECT ticket\_no FROM bookings WHERE train\_id = p\_train\_id AND date\_of\_journey = p\_date\_of\_journey;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

CREATE TEMPORARY TABLE IF NOT EXISTS temp\_passengers LIKE passenger;

OPEN c1;

read\_loop: LOOP

FETCH c1 INTO ticketno;

IF done THEN

LEAVE read\_loop;

END IF;

INSERT INTO temp\_passengers SELECT \* FROM passenger WHERE ticket\_no = ticketno;

END LOOP;

CLOSE c1;

SELECT \* FROM temp\_passengers;

DROP TEMPORARY TABLE IF EXISTS temp\_passengers;

END

3.

CREATE DEFINER=`root`@`localhost` PROCEDURE `get\_trains`(IN input\_origin VARCHAR(50), IN input\_destination VARCHAR(50),p\_date DATE)

BEGIN

SELECT train\_id, origin, destination, arrival,departure,date\_of\_journey

FROM train\_tt

WHERE origin = input\_origin AND destination = input\_destination AND date\_of\_journey=p\_date;

END

4.

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert\_cancellation`(IN ticket\_no INT, IN p\_email VARCHAR(50))

BEGIN

DECLARE fees INT;

-- Call the cancellation\_fee function correctly using the input parameter ticket\_no

SELECT cancellation\_fee(ticket\_no) INTO fees;

-- Use the input parameter ticket\_no in the INSERT statement

INSERT INTO cancellation (ticket\_no, cancellation\_date, charges)

VALUES(ticket\_no, CURDATE(), fees);

-- Ensure user\_profile table exists and the email exists in it

UPDATE user\_profile

SET money = money + fees

WHERE email = p\_email;

END

5.

CREATE DEFINER=`root`@`localhost` PROCEDURE `register\_new\_user`(

IN email\_ VARCHAR(50),

IN fname VARCHAR(50),

IN lname VARCHAR(50),

IN passwd VARCHAR(50),

IN conf\_passwd VARCHAR(50)

)

BEGIN

DECLARE count INT;

SELECT COUNT(\*) INTO count FROM user\_login WHERE email = email\_;

IF count=0 THEN

IF passwd = conf\_passwd THEN

INSERT INTO user\_login (email, first\_name, last\_name, passwd) VALUES (email\_, fname, lname, passwd);

INSERT INTO user\_profile (email, first\_name, last\_name)

SELECT email, first\_name, last\_name FROM user\_login WHERE email = email\_;

ELSE

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Passwords do not match.';

END IF;

ELSE

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Email already exists.';

END IF;

END

6.

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_train\_arrival`(IN id varchar(6), IN journey\_date DATE, IN arrival\_time TIME)

BEGIN

UPDATE train\_tt

SET arrival = arrival\_time

WHERE train\_id = id AND date\_of\_journey = journey\_date;

END

7.

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_train\_departure`(IN id varchar(6), IN journey\_date DATE, IN departure\_time TIME)

BEGIN

UPDATE train\_tt

SET departure = departure\_time

WHERE train\_id = id AND date\_of\_journey = journey\_date;

END

8.

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_train\_time`(IN id varchar(6), IN journey\_date DATE, IN arrival\_time TIME)

BEGIN

UPDATE train\_tt

SET arrival = arrival\_time

WHERE train\_id = id AND date\_of\_journey = journey\_date;

END

Functions

1.

CREATE DEFINER=`root`@`localhost` FUNCTION `calculate\_employee\_bonus`(id varchar(5)) RETURNS int

DETERMINISTIC

BEGIN

DECLARE salary\_amount INT;

DECLARE bonus\_amount INT;

SELECT j.salary into salary\_amount

FROM employee e

JOIN jobs j ON e.job\_type = j.job\_type

WHERE e.emp\_id = id;

IF salary\_amount > 50000 THEN

SET bonus\_amount = salary\_amount \* 0.1;

ELSEIF salary\_amount > 25000 THEN

SET bonus\_amount = salary\_amount \* 0.2;

ELSE

SET bonus\_amount = salary\_amount \* 0.3;

END IF;

RETURN bonus\_amount;

END

2.

CREATE DEFINER=`root`@`localhost` FUNCTION `calculate\_train\_revenue`(p\_train\_id varchar(6)) RETURNS int

DETERMINISTIC

BEGIN

DECLARE total\_revenue INT;

SELECT SUM(fare)

INTO total\_revenue

FROM bookings

WHERE train\_id = p\_train\_id;

RETURN total\_revenue;

END

3.

CREATE DEFINER=`root`@`localhost` FUNCTION `cancellation\_fee`(ticketno int) RETURNS int

DETERMINISTIC

BEGIN

DECLARE date DATE;

DECLARE no\_of\_days INT;

DECLARE base\_fee INT;

DECLARE price INT;

BEGIN

SET base\_fee = 50;

SELECT fare INTO price

FROM bookings

WHERE ticket\_no=ticketno;

SELECT DATEDIFF(date\_of\_journey, CURDATE()) INTO no\_of\_days

FROM bookings

WHERE ticket\_no=ticketno;

IF no\_of\_days>=30 THEN

RETURN base\_fee;

ELSEIF no\_of\_days>=15 THEN

RETURN price/10;

ELSE

RETURN price/5;

END IF;

RETURN 1;

END;

END

4.

CREATE DEFINER=`root`@`localhost` FUNCTION `get\_available\_seats`(id varchar(6), class VARCHAR(2)) RETURNS int

DETERMINISTIC

BEGIN

DECLARE total INT;

DECLARE occupied INT;

SELECT total\_seats INTO total

FROM seats

WHERE train\_id = id AND class\_name = class;

-- Subtract the number of booked seats for the given train\_id and class

SELECT COUNT(\*) INTO occupied

FROM confirmed\_seats

WHERE train\_id = id AND class\_name = class;

RETURN total - occupied;

END

5.

CREATE DEFINER=`root`@`localhost` FUNCTION `get\_max\_seat\_number`(p\_train\_id varchar(6), p\_class\_name VARCHAR(2), p\_date\_of\_journey DATE) RETURNS int

DETERMINISTIC

BEGIN

DECLARE max\_seat\_number INT;

SELECT COALESCE(MAX(seat\_no), 0) INTO max\_seat\_number

FROM confirmed\_seats

WHERE train\_id = p\_train\_id AND class\_name = p\_class\_name AND date\_of\_journey = p\_date\_of\_journey;

RETURN max\_seat\_number;

END

Triggers

1.

CREATE DEFINER=`root`@`localhost` TRIGGER `insert\_passenger` AFTER INSERT ON `bookings` FOR EACH ROW BEGIN

INSERT INTO passenger (ticket\_no, pass\_first\_name, pass\_last\_name, train\_id, class\_name, gender, age, contact, aadhar\_ID)

SELECT NEW.ticket\_no, user\_profile.first\_name, user\_profile.last\_name, NEW.train\_id,

NEW.class\_name, user\_profile.gender, TIMESTAMPDIFF(YEAR,user\_profile.dob,CURDATE()), user\_profile.contact, user\_profile.aadhar\_ID

FROM user\_profile

WHERE user\_profile.email = NEW.email;

END

2.

CREATE DEFINER=`root`@`localhost` TRIGGER `cancel\_booking` AFTER INSERT ON `cancellation` FOR EACH ROW BEGIN

DECLARE trainid VARCHAR(6);

DECLARE class VARCHAR(2);

DECLARE seat INT;

DECLARE journey\_date DATE;

DELETE FROM passenger WHERE ticket\_no = NEW.ticket\_no;

UPDATE bookings SET status = 'C' WHERE ticket\_no=NEW.ticket\_no;

SELECT train\_id, class\_name, seat\_no, date\_of\_journey INTO trainid, class, seat, journey\_date

FROM bookings

WHERE ticket\_no = NEW.ticket\_no;

DELETE FROM confirmed\_seats

WHERE train\_id = trainid AND class\_name = class AND seat\_no = seat AND date\_of\_journey = journey\_date;

END

3.

CREATE DEFINER=`root`@`localhost` TRIGGER `insert\_price\_history` AFTER UPDATE ON `seats` FOR EACH ROW INSERT INTO fare\_history (

train\_id,

class\_name,

old\_fare,

new\_fare,

price\_change\_date,

price\_change\_time

)

VALUES (

NEW.train\_id,

NEW.class\_name,

OLD.fare,

NEW.fare,

CURDATE(),

CURTIME()

)

4.

CREATE DEFINER=`root`@`localhost` TRIGGER `train\_delayed\_trigger` AFTER UPDATE ON `train\_tt` FOR EACH ROW BEGIN

IF NEW.arrival <> OLD.arrival OR NEW.departure <> OLD.departure THEN

INSERT INTO train\_delayed (train\_id, date\_of\_journey, arrival, arrival\_delayed, departure, departure\_delayed,

date\_changed,time\_changed)

VALUES (NEW.train\_id, NEW.date\_of\_journey, OLD.arrival, TIMESTAMPDIFF(MINUTE, OLD.arrival, NEW.arrival), OLD.departure,

TIMESTAMPDIFF(MINUTE, OLD.departure, NEW.departure),CURDATE(),CURTIME());

END IF;

END

5.

CREATE DEFINER=`root`@`localhost` TRIGGER `check\_user\_age` BEFORE UPDATE ON `user\_profile` FOR EACH ROW BEGIN

DECLARE birthdate DATE;

DECLARE age INT;

SET birthdate = NEW.dob;

SET age = TIMESTAMPDIFF(YEAR,birthdate,CURDATE());

IF age < 18 THEN

SIGNAL SQLSTATE '45001'

SET MESSAGE\_TEXT = 'You must be 18 or older to register and make bookings.';

END IF;

END