TIME TO *Travel*

BOOK NOW

5. WRITE RELEVANT QUERIES TO ACCESS THE DATA FROM TABLES --
RETRIEVE ALL USER DATA:
SELECT * FROM USER;

-- RETRIEVE USER DATA FOR A SPECIFIC USER BY USERID:
SELECT * FROM USER WHERE USERID = '5';

-- RETRIEVE ALL DESTINATION DATA:
SELECT * FROM DESTINATION;

-- RETRIEVE DESTINATION DATA FOR A SPECIFIC DESTINATION BY DESTINATIONID:
SELECT * FROM DESTINATION WHERE DESTINATIONID = '6';

-- RETRIEVE ALL TOUR DATA:
SELECT * FROM TOUR;

-- RETRIEVE TOUR DATA FOR A SPECIFIC TOUR BY TOURID:
SELECT * FROM TOUR WHERE TOURID = '107';

-- RETRIEVE ALL BOOKING DATA:
SELECT * FROM BOOKING;

-- RETRIEVE BOOKING DATA FOR A SPECIFIC BOOKING BY BOOKINGID:
SELECT * FROM BOOKING WHERE BOOKINGID = '18';

-- RETRIEVE ALL PAYMENT DATA:
SELECT * FROM PAYMENT;

-- RETRIEVE PAYMENT DATA FOR A SPECIFIC PAYMENT BY PAYMENTID:
SELECT * FROM PAYMENT WHERE PAYMENTID = '16';

-- RETRIEVE ALL REVIEW DATA:

```
SELECT * FROM REVIEW;

-- RETRIEVE REVIEW DATA FOR A SPECIFIC REVIEW BY REVIEWID:
SELECT * FROM REVIEW WHERE REVIEWID = '16';

-- RETRIEVE ALL STAFF DATA:
SELECT * FROM STAFF;

-- RETRIEVE STAFF DATA FOR A SPECIFIC STAFF MEMBER BY
STAFFID:
SELECT * FROM STAFF WHERE STAFFID = '6';

-- RETRIEVE ALL ACTIVITY DATA:
SELECT * FROM ACTIVITY;

-- RETRIEVE ACTIVITY DATA FOR A SPECIFIC ACTIVITY BY
ACTIVITYID:
SELECT * FROM ACTIVITY WHERE ACTIVITYID = '13';
```

```
6. WRITE AT LEAST 3 TRIGGERS – BEFORE MODIFY AND UPDATE AND
AFTER DELETE
DELIMITER $$
CREATE TRIGGER `BOOKING_BEFORE_UPDATE` BEFORE UPDATE
ON `BOOKING`
FOR EACH ROW
BEGIN
  IF NEW.USERID != OLD.USERID THEN
    SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'UNAUTHORIZED UPDATE: USERID CANNOT
BE CHANGED.';
  END IF;
END$$

CREATE TRIGGER `PAYMENT_BEFORE_MODIFY` BEFORE INSERT ON
`PAYMENT`
FOR EACH ROW
```

```
BEGIN
  IF NEW.`PAYMENT TYPE` NOT IN ('CARD', 'UPI', 'NEFT','CASH') THEN
    SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'INVALID PAYMENT TYPE.';
  END IF;
END$$



CREATE TRIGGER `DESTINATION_AFTER_DELETE` AFTER DELETE ON
`DESTINATION`
FOR EACH ROW
BEGIN
  UPDATE TOUR SET DESTINATIONID = NULL WHERE DESTINATIONID
= OLD.DESTINATIONID;
END$$
```

```
 7. CREATE AT LEAST 6 VIEWS ON TABLES OF YOUR CHOICE
CREATE VIEW `BOOKING_WITH_USER_INFO` AS
SELECT B.BOOKINGID, B.USERID, U.FNAME, U.LNAME, U.EMAIL,
U.PHONE, B.TOURID, B.`BOOKING DATE`, B.TOTALCOST
FROM BOOKING B
INNER JOIN USER U ON B.USERID = U.USERID;

CREATE VIEW `TOUR_WITH_DESTINATION_INFO` AS
SELECT T.TOURID, T.NAME, T.`DURATIONDAYS`, T.PRICE, D.NAME
AS DESTINATIONNAME, D.COUNTRY, D.DESCRIPTION
FROM DESTINATIONTOUR T
INNER JOIN DESTINATION D ON T.DESTINATIONID =
D.DESTINATIONID;

CREATE VIEW `PAYMENT_WITH_BOOKING_INFO` AS
SELECT P.PAYMENTID, P.BOOKINGID, B.USERID, B.TOURID,
B.`BOOKING DATE`, B.TOTALCOST, P.`PAYMENT TYPE`
```

```
FROM PAYMENT P
INNER JOIN BOOKING B ON P.BOOKINGID = B.BOOKINGID;

CREATE VIEW `USER_WITH_BOOKING_COUNT` AS
SELECT U.USERID, U.FNAME, U.LNAME, COUNT(B.BOOKINGID) AS
BOOKINGCOUNT
FROM USER U
LEFT JOIN BOOKING B ON U.USERID = B.USERID
GROUP BY U.USERID;
```

```
CREATE VIEW `DESTINATION_WITH_AVG_RATING` AS
SELECT D.DESTINATIONID, D.NAME, D.COUNTRY, D.DESCRIPTION,
AVG(R.RATING) AS AVGRATING
FROM DESTINATION D
LEFT JOIN TOUR T ON D.DESTINATIONID = T.DESTINATIONID
LEFT JOIN REVIEW R ON T.TOURID = R.TOURID
GROUP BY D.DESTINATIONID;
```

```
CREATE VIEW `TOUR_WITH_REVENUE` AS
SELECT T.TOURID, COUNT(B.BOOKINGID) AS BOOKINGCOUNT,
SUM(B.TOTALCOST) AS TOTALREVENUE
FROM DESTINATIONTOUR T
LEFT JOIN BOOKING B ON T.TOURID = B.TOURID
GROUP BY T.TOURID;
8. WRITE AT LEAST 3 FUNCTIONS AND PROCEDURE ON TABLES OF
YOUR CHOICE
DELIMITER $$
CREATE FUNCTION `CALCULATE_BOOKING_COST`(BOOKING_ID INT)
```

```
RETURNS DECIMAL(10,2)
BEGIN
DECLARE TOTAL_COST DECIMAL(10,2);
SELECT (B.TOTALCOST + (B.TOTALCOST * 0.10)) INTO TOTAL_COST --
ADD 10% TAX
FROM BOOKING B
WHERE B.BOOKINGID = BOOKING_ID;
RETURN TOTAL_COST;
END$$
```

```
CREATE PROCEDURE `UPDATE_TOUR_PRICE`(IN TOUR_ID INT, IN
NEW_PRICE DECIMAL(10,2))
BEGIN
UPDATE TOUR SET PRICE = NEW_PRICE WHERE TOURID = TOUR_ID; -
- UPDATE TOUR PRICE
UPDATE BOOKING SET TOTALCOST = (TOTALCOST - (TOTALCOST *
0.1) + (NEW_PRICE * 1.1)) -- UPDATE BOOKING COSTS
WHERE TOURID = TOUR_ID;
END$$
```

```
 CREATE FUNCTION `CALCULATE_TOUR_RATING`(TOUR_ID INT)
RETURNS DECIMAL(2,1)
BEGIN
DECLARE AVG_RATING DECIMAL(2,1);
SELECT AVG(R.RATING) INTO AVG_RATING
FROM REVIEW R
WHERE R.TOURID = TOUR_ID;
RETURN AVG_RATING;
END$$
```

```
 9. WRITE AT LEAST 2 CURSORS ON TABLES OF YOUR CHOICE
DELIMITER //

CREATE PROCEDURE CALCULATE_BOOKING_COSTS(IN USER_ID INT)
BEGIN
    DECLARE BOOKING_ID INT;
    DECLARE TOTAL_COST DECIMAL(10,2);
    DECLARE CUR CURSOR FOR
        SELECT BOOKINGID, (TOTALCOST + (TOTALCOST * 0.10)) AS
TOTAL_COST -- ADD 10% TAX
        FROM BOOKING
        WHERE USERID = USER_ID;

    OPEN CUR;
```

```
FETCH NEXT FROM CUR INTO BOOKING_ID, TOTAL_COST;

    WHILE FOUND_ROWS() DO
        -- DO SOMETHING WITH BOOKING_ID AND TOTAL_COST
        -- E.G. UPDATE A REPORT TABLE OR SEND AN EMAIL TO THE
USER

        FETCH NEXT FROM CUR INTO BOOKING_ID, TOTAL_COST;
    END WHILE;

    CLOSE CUR;
END //


DELIMITER //

CREATE FUNCTION CALCULATE_TOUR_REVENUE()
RETURNS DECIMAL(10,2)
BEGIN
    DECLARE TOUR_ID INT;
    DECLARE TOTAL_REVENUE DECIMAL(10,2);
    DECLARE CUR CURSOR FOR
        SELECT T.TOURID, SUM(B.TOTALCOST) AS TOTAL_REVENUE
        FROM TOUR T
        JOIN BOOKING B ON B.TOURID = T.TOURID
        GROUP BY T.TOURID;

    SET TOTAL_REVENUE = 0;

    OPEN CUR;

    FETCH NEXT FROM CUR INTO TOUR_ID, TOTAL_REVENUE;
```

```
WHILE FOUND_ROWS() DO
        -- DO SOMETHING WITH TOUR_ID AND TOTAL_REVENUE
        -- E.G. UPDATE A REPORT TABLE OR GENERATE A SALES
REPORT
        SET TOTAL_REVENUE = TOTAL_REVENUE + TOTAL_REVENUE;

        FETCH NEXT FROM CUR INTO TOUR_ID, TOTAL_REVENUE;
    END WHILE;

    CLOSE CUR;

    RETURN TOTAL_REVENUE;
END //
```

10. CREATE USERS AND ASSIGN DIFFERENT ROLES TO DIFFERENT USERS.

```
CREATE USER 'E1'@'LOCALHOST' IDENTIFIED BY 'PASSWORD1';
CREATE USER 'E2'@'LOCALHOST' IDENTIFIED BY 'PASSWORD2';

CREATE ROLE MANAGER;
CREATE ROLE RECEPTIONIST;


GRANT ALL PRIVILEGES ON CIE3.* TO MANAGER;
GRANT SELECT, INSERT, UPDATE ON CIE3.BOOKING TO
RECEPTIONIST;
```

GROUP MEMBERS NAME :

KISHAL SHAH - 31

ASHUTOSH DESHMUKH - 36

TUSHAR CHEDE - 39

ATTHARVA DAGA - 25