Tour & Travel Website Database: Implementation & Output

1. Database Schema (Input)

```
les
ries (country_code CHAR(2) PRIMARY KEY, country_name VARCHAR(100) NOT NULL UNIQUE);
 (user_id INT PRIMARY KEY AUTO_INCREMENT, email VARCHAR(100) NOT NULL UNIQUE, password_hash VARCHAR(255) NOT NULL, role

 & Inventory
nations (destination_id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(100) NOT NULL, country_code CHAR(2) NOT NULL, FOREI
 (tour_id INT PRIMARY KEY AUTO_INCREMENT, title VARCHAR(255) NOT NULL, destination_id INT NOT NULL, description TEXT, du
vailability (availability_id INT PRIMARY KEY AUTO_INCREMENT, tour_id INT NOT NULL, start_date DATE NOT NULL, end_date DA

ction & Review Tables
tions (promo_id INT PRIMARY KEY AUTO_INCREMENT, promo_code VARCHAR(50) NOT NULL UNIQUE, discount_percentage DECIMAL(5, 2
ngs (booking_id INT PRIMARY KEY AUTO_INCREMENT, user_id INT NOT NULL, availability_id INT NOT NULL, promo_id INT NULL, b
nts (payment_id INT PRIMARY KEY AUTO_INCREMENT, booking_id INT NOT NULL, payment_date TIMESTAMP DEFAULT CURRENT_TIMESTAM
ws (review_id INT PRIMARY KEY AUTO_INCREMENT, booking_id INT NOT NULL UNIQUE, user_id INT NOT NULL, tour_id INT NOT NULL
```

2. Data Integrity & Business Logic (Input)

Triggers: Automated Inventory and Status Management

```sql
-- Trigger 1: Increment/decrement seats_booked when a booking is confirmed or cancelled.
DELIMITER $$
CREATE TRIGGER after_booking_status_change
AFTER UPDATE ON Bookings
FOR EACH ROW
BEGIN
    -- User confirms their booking
    IF NEW.status = 'Confirmed' AND OLD.status <> 'Confirmed' THEN
        UPDATE TourAvailability SET seats_booked = seats_booked + NEW.number_of_travelers WHERE availability_id = NEW
    -- User cancels their confirmed booking
    ELSEIF NEW.status = 'Cancelled' AND OLD.status = 'Confirmed' THEN
        UPDATE TourAvailability SET seats_booked = seats_booked - NEW.number_of_travelers WHERE availability_id = NEW
    END IF;
END$$
DELIMITER ;

-- Trigger 2: Automatically update tour status to 'SoldOut' or 'Open'.
DELIMITER $$
CREATE TRIGGER after_seats_booked_updated
AFTER UPDATE ON TourAvailability
FOR EACH ROW
BEGIN
    IF NEW.seats_booked >= NEW.max_capacity THEN
        UPDATE TourAvailability SET status = 'SoldOut' WHERE availability_id = NEW.availability_id;
    ELSEIF NEW.seats_booked < NEW.max_capacity AND NEW.status = 'SoldOut' THEN
         UPDATE TourAvailability SET status = 'Open' WHERE availability_id = NEW.availability_id;
    END IF;
END$$
DELIMITER ;
```

Stored Procedures: Encapsulated Business Processes

```
rocedure 1: Atomically create a new booking with validation.
IITER $$
E PROCEDURE CreateBooking(...) /* Full code from previous example */ $$
IITER ;

rocedure 2: Confirm a booking after successful payment.
IITER $$
E PROCEDURE ConfirmBookingAndPayment(
N p_booking_id INT,
N p_transaction_id VARCHAR(255),
N p_payment_amount DECIMAL(10, 2)


ECLARE booking_status VARCHAR(20);
TART TRANSACTION;
ELECT status INTO booking_status FROM Bookings WHERE booking_id = p_booking_id;
F booking_status = 'Pending' THEN
   INSERT INTO Payments (booking_id, amount, transaction_id, status) VALUES (p_booking_id, p_payment_amount, p_transact
   UPDATE Bookings SET status = 'Confirmed' WHERE booking_id = p_booking_id;
   COMMIT;
LSE
   ROLLBACK;
   SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Booking is not in Pending state.';
ND IF;

IITER ;

rocedure 3: Add a review, but only for a completed tour.
IITER $$
E PROCEDURE AddReview(
N p_booking_id INT,
N p_rating INT,
N p_comment TEXT
```

```
ECLARE booking_status VARCHAR(20);
ECLARE v_user_id, v_tour_id INT;
ELECT status, user_id, ta.tour_id INTO booking_status, v_user_id, v_tour_id
ROM Bookings b JOIN TourAvailability ta ON b.availability_id = ta.availability_id
HERE b.booking_id = p_booking_id;
F booking_status = 'Completed' THEN
   INSERT INTO Reviews(booking_id, user_id, tour_id, rating, comment) VALUES(p_booking_id, v_user_id, v_tour_id, p_rati
LSE
   SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Reviews can only be added for completed tours.';
ND IF;

IITER ;
```

3. Search & Reporting Views (Input)

```
-- View 1: A comprehensive, searchable list of all available tours.
CREATE OR REPLACE VIEW V_AvailableTours AS
SELECT
    t.tour_id, ta.availability_id, t.title, t.description, d.name AS destination, c.country_name,
    t.duration_days, ta.start_date, ta.end_date,
    (t.base_price * ta.price_modifier) AS price_per_person,
    (ta.max_capacity - ta.seats_booked) AS seats_available
FROM Tours t
JOIN Destinations d ON t.destination_id = d.destination_id
JOIN Countries c ON d.country_code = c.country_code
JOIN TourAvailability ta ON t.tour_id = ta.tour_id
WHERE ta.status = 'Open' AND ta.start_date > CURDATE();

-- View 2: Aggregated tour data including average rating.
CREATE OR REPLACE VIEW V_TourRatings AS
SELECT t.tour_id, t.title, AVG(r.rating) as average_rating, COUNT(r.review_id) as review_count
FROM Tours t
```

```
LEFT JOIN Reviews r ON t.tour_id = r.tour_id
GROUP BY t.tour_id, t.title;
```

4. Seed Data (Input)

```
ta is assumed to be loaded for the following queries to work.
O Countries (country_code, country_name) VALUES ('FR','France'), ('IT','Italy'), ('JP','Japan');
O Users (user_id, email, password_hash) VALUES (1,'alice@example.com','h1'), (2,'bob@example.com','h2');
O Promotions (promo_id, promo_code, discount_percentage, valid_from, valid_to) VALUES (1, 'EARLYBIRD10', 10.00, '2025-0
O Destinations (destination_id, name, country_code) VALUES (1,'Paris','FR'), (2,'Rome','IT'), (3,'Tokyo','JP');
O Tours (tour_id, title, destination_id, description, duration_days, base_price) VALUES
risian Wonders',1,'A romantic journey through Paris.',5,1200.00),
cient Rome Explorer',2,'Explore the Colosseum and ancient ruins.',7,1800.00),
kyo Neon Nights',3,'Experience the vibrant city life of Tokyo.',10,3000.00);
O TourAvailability (availability_id, tour_id, start_date, end_date, max_capacity, seats_booked, price_modifier, status)
2025-10-20','2025-10-24',20,18,1.10,'Open'),
2025-09-15','2025-09-21',15,10,1.00,'Open'),
2026-04-01','2026-04-10',12,0,1.50,'Open'),
2025-11-10','2025-11-14',20,5,1.00,'Open');
O Bookings (booking_id, user_id, availability_id, promo_id, number_of_travelers, final_total_price, status) VALUES
1, 2, NULL, 2, 3600.00, 'Completed');
O Reviews (review_id, booking_id, user_id, tour_id, rating, comment) VALUES (1, 101, 1, 2, 5, 'Amazing trip, the guide
```

5. Queries & Outputs

5.1 Search & Discovery Queries

1. Find tours within a price range (€1000 - €2000 per person)

```
SELECT title, destination, price_per_person, start_date FROM V_AvailableTours WHERE price_per_person BETWEEN 100
```

Output:

title	destination	price_per_person	start_date
Parisian Wonders	Paris	1320.00	2025-10-20
Ancient Rome Explorer	Rome	1800.00	2025-09-15
Parisian Wonders	Paris	1200.00	2025-11-10

2. Find "last minute deals" (tours starting in the next 60 days)

```
CT title, start_date, seats_available FROM V_AvailableTours WHERE start_date BETWEEN CURDATE() AND DATE_ADD(CURDATE
```

Output:

title	start_date	seats_available
Ancient Rome Explorer	2025-09-15	5

3. Search for tours by keyword 'explore' in the description

```
SELECT title, destination, duration_days FROM V_AvailableTours WHERE description LIKE '%explore%';
```

Output:

title	destination	duration_days
Ancient Rome Explorer	Rome	7

4. List all tours with their average user rating

```
SELECT title, average_rating, review_count FROM V_TourRatings ORDER BY average_rating DESC;
```

Output:

title	average_rating	review_count
Ancient Rome Explorer	5.0000	1
Parisian Wonders	NULL	0
Tokyo Neon Nights	NULL	0

5.2 Transactional & Management Queries

1. Create a new booking for 2 people on the almost-full Parisian Wonders tour

```
CALL CreateBooking(2, 1, 2, 'EARLYBIRD10');
```

Output:

> Action successful. A new row is inserted into `Bookings`. We can verify with a SELECT.

booking_id	user_id	availability_id	final_total_price	status
102	2	1	2376.00	Pending

2. Confirm the booking after payment

```
CALL ConfirmBookingAndPayment(102, 'txn_xyz789', 2376.00);
```

Output:

> Action successful. The `Bookings` status is updated to `Confirmed`, a `Payments` record is created, and the `after_booking_status_change` trigger fires. This also fires the `after_seats_booked_updated` trigger, setting the tour to 'SoldOut'.

availability_id	tour_id	seats_booked	max_capacity	status
1	1	20	20	SoldOut

3. View a specific user's complete booking history

```
SELECT b.booking_id, t.title, b.status, b.final_total_price, b.booking_date
FROM Bookings b
JOIN TourAvailability ta ON b.availability_id = ta.availability_id
JOIN Tours t ON ta.tour_id = t.tour_id
WHERE b.user_id = 1;
```

Output:

booking_id	title	status	final_total_price	booking_date
101	Ancient Rome Explorer	Completed	3600.00	2025-08-21 14:30:00

4. Attempt to add a review for a tour that is not yet 'Completed'

```
-- User Bob (ID 2) tries to review his 'Confirmed' but not yet 'Completed' booking (ID 102)
CALL AddReview(102, 5, 'Looking forward to it!');
```

Output:

ERROR 1644 (45000): Reviews can only be added for completed tours.

5.3 Analytics & Reporting Queries

1. Sales Performance by Destination

```
SELECT d.name as destination, COUNT(b.booking_id) as total_bookings, SUM(b.final_total_price) as total_revenue
FROM Bookings b
JOIN TourAvailability ta ON b.availability_id = ta.availability_id
JOIN Tours t ON ta.tour_id = t.tour_id
JOIN Destinations d ON t.destination_id = d.destination_id
```

```
WHERE b.status IN ('Confirmed', 'Completed')
GROUP BY d.name ORDER BY total_revenue DESC;
```

Output:

destination	total_bookings	total_revenue
Rome	1	3600.00
Paris	1	2376.00

2. Most Popular Tours (by number of travelers booked)

```
SELECT t.title, SUM(b.number_of_travelers) as total_travelers
FROM Bookings b
JOIN TourAvailability ta ON b.availability_id = ta.availability_id
JOIN Tours t ON ta.tour_id = t.tour_id
WHERE b.status IN ('Confirmed', 'Completed')
GROUP BY t.title ORDER BY total_travelers DESC;
```

Output:

title	total_travelers
Ancient Rome Explorer	2
Parisian Wonders	2

3. Promotion Code Usage Statistics

```
SELECT p.promo_code, COUNT(b.booking_id) as times_used, SUM(b.final_total_price) as revenue_generated
FROM Promotions p
JOIN Bookings b ON p.promo_id = b.promo_id
WHERE b.status IN ('Confirmed', 'Completed')
GROUP BY p.promo_code;
```

Output:

promo_code	times_used	revenue_generated
EARLYBIRD10	1	2376.00

4. Occupancy Rate for all Tour Departures

```
:le, ta.start_date, ta.seats_booked, ta.max_capacity, CONCAT(ROUND((ta.seats_booked / ta.max_capacity) * 100, 2), '
iilability ta
: ON ta.tour_id = t.tour_id
 start_date;
```

Output:

title	start_date	seats_booked	max_capacity	occupancy_rate
Ancient Rome Explorer	2025-09-15	10	15	66.67%
Parisian Wonders	2025-10-20	20	20	100.00%
Parisian Wonders	2025-11-10	5	20	25.00%

title	start_date	seats_booked	max_capacity	occupancy_rate
Tokyo Neon Nights	2026-04-01	0	12	0.00%