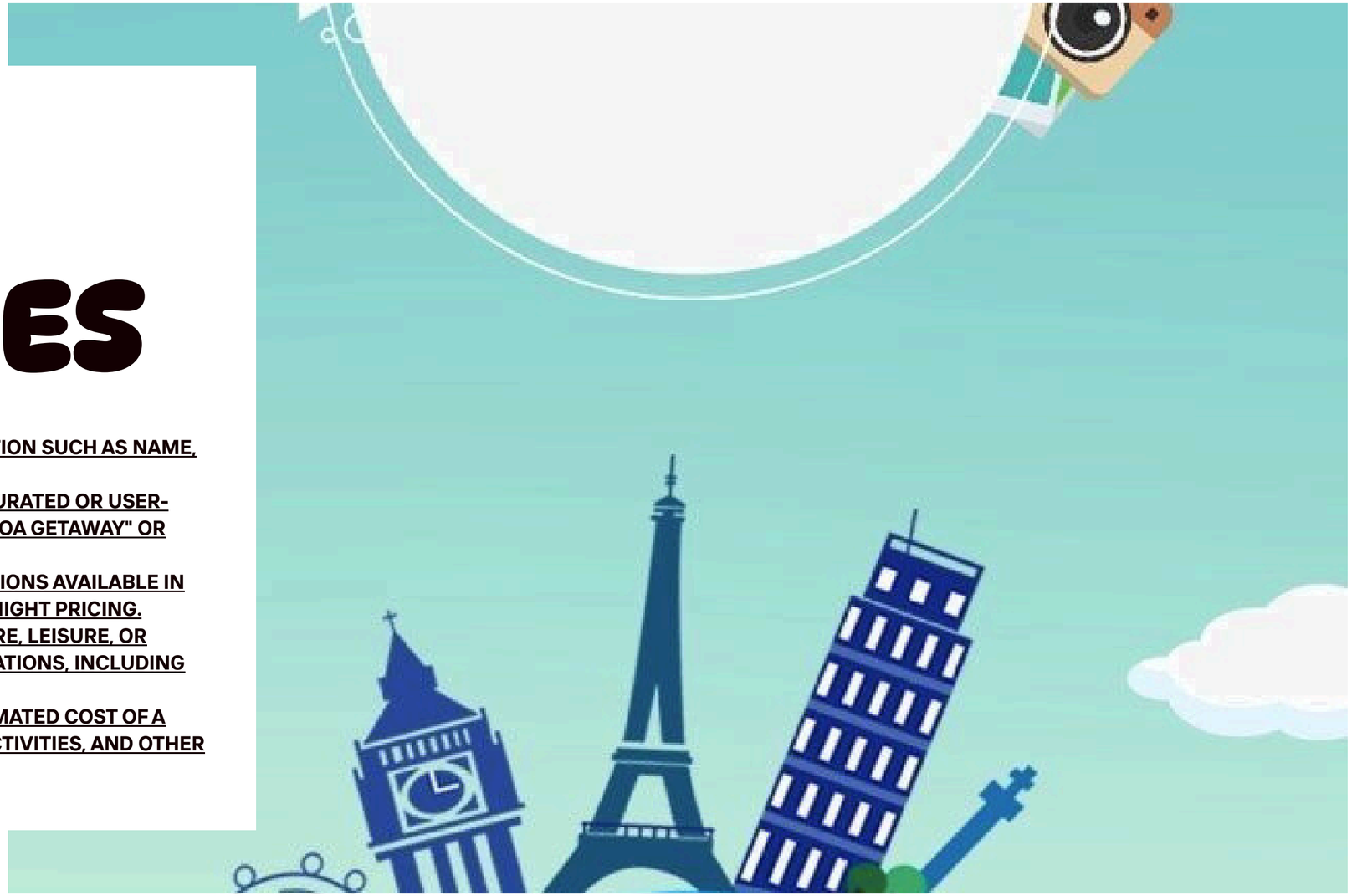

Tour Package
Planner
POOJA TANWAR

Introduction

The Tour Package Planner with Price Estimator is allowing users to create customized travel packages by selecting destinations, hotels, and activities, while getting real-time cost estimates. This project helps tour companies automate pricing, optimize user experience, and ensure accurate cost planning.

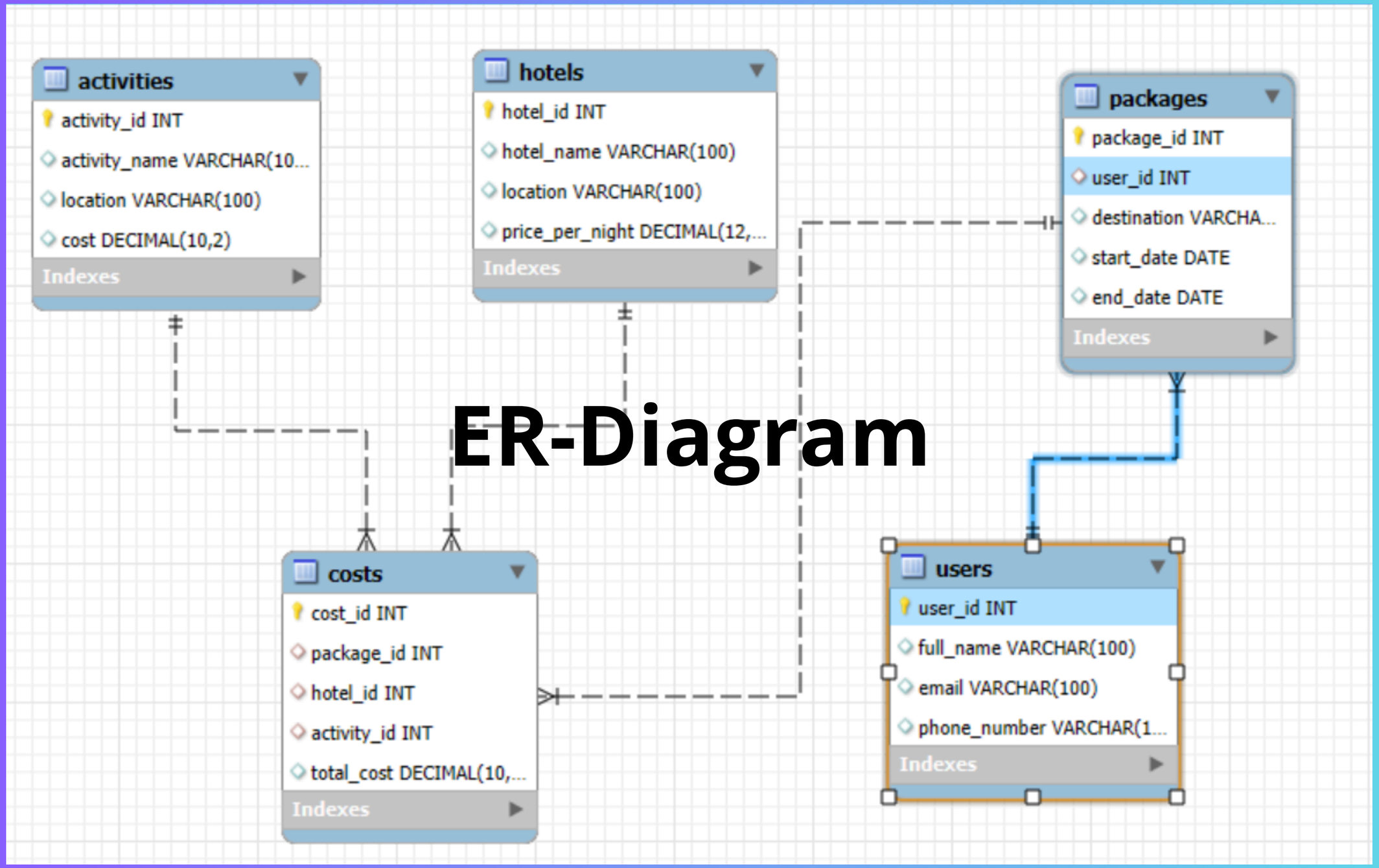
KEY FEATURES

- 🚹🚺 USERS (USERS): STORES TRAVELLER INFORMATION SUCH AS NAME, EMAIL, AND CONTACT DETAILS.
- 🌍 TRAVEL PACKAGES (PACKAGES): CONTAINS CURATED OR USER-DESIGNED TRAVEL PACKAGES LIKE "ROMANTIC GOA GETAWAY" OR "ADVENTURE IN MANALI".
- 🏨 HOTELS (HOTELS): LISTS ACCOMMODATION OPTIONS AVAILABLE IN DIFFERENT DESTINATIONS, ALONG WITH PER-NIGHT PRICING.
- 🎯 ACTIVITIES (ACTIVITIES): STORES ADVENTURE, LEISURE, OR SIGHTSEEING ACTIVITIES LINKED TO SPECIFIC LOCATIONS, INCLUDING THEIR INDIVIDUAL COSTS.
- 💰 COST DETAILS (COSTS): RECORDS TOTAL ESTIMATED COST OF A PACKAGE BY COMBINING HOTEL STAYS, SELECTED ACTIVITIES, AND OTHER PRICING ELEMENTS.



WHAT THIS SYSTEM CAN DO:

- 🧳 ALLOW USERS TO CREATE PERSONALIZED TOUR PACKAGES BY SELECTING DESTINATIONS, HOTELS AND ACTIVITIES.
- 💸 INSTANTLY CALCULATE AND DISPLAY TOTAL ESTIMATED COST BASED ON SELECTED COMPONENTS.
- 🏨 SHOW AVAILABLE HOTELS AND ACTIVITIES IN CHOSEN LOCATIONS.
- 📊 GENERATE DETAILED COST BREAKDOWNS INCLUDING HOTEL CHARGES, ACTIVITY FEES, AND TOTAL PACKAGE PRICE.
- 📂 MAINTAIN A HISTORY OF PAST PACKAGES DESIGNED OR BOOKED BY USERS FOR REFERENCE AND INSIGHTS.
- 🔍 HELP TRAVEL COMPANIES MANAGE OFFERINGS, PRICING, AND POPULAR PACKAGE TRENDS USING SQL QUERIES.


activities
activity_id INT
activity_name VARCHAR(10...
location VARCHAR(100)
cost DECIMAL(10,2)
Indexes
hotels
hotel_id INT
hotel_name VARCHAR(100)
location VARCHAR(100)
price_per_night DECIMAL(12,...
Indexes
packages
package_id INT
user_id INT
destination VARCHA...
start_date DATE
end_date DATE
Indexes
ER-Diagram
costs
cost_id INT
package_id INT
hotel_id INT
activity_id INT
total_cost DECIMAL(10,...
Indexes
users
user_id INT
full_name VARCHAR(100)
email VARCHAR(100)
phone_number VARCHAR(1...
Indexes

Let's start building!

Step 1: Setting Up the Database Schema

Creating the structure where data will be stored and organized — like building the foundation of a system before you start entering data.

Create Database

```
CREATE DATABASE TOURPLANNER;
USE TOURPLANNER;
```

Create Table

```
Create Table Users (
User_id Int Primary Key,
name Varchar(100),
email varchar(100) unique);
(To Describe Structure of Tables)
desc Users;
```

Users Table

Field	Type	Null	Key	Default	Extra
user_id	int	NO	PRI	NULL	
User_name	varchar(100)	YES		NULL	
email	varchar(100)	YES	UNI	NULL	

```
Create Table Packages (
package_id Int Primary Key,
user_id Int,
destination  varchar(100),
start_date Date,
end_date Date,
Foreign Key (user_id) references Users(user_id));
desc Packages;
```

Packages Table

Field	Type	Null	Key	Default	Extra
package_id	int	NO	PRI	NULL	
user_id	int	YES	MUL	NULL	
destination	varchar(100)	YES		NULL	
start_date	date	YES		NULL	
end_date	date	YES		NULL	

```
Create Table Hotels (
hotel_id Int Primary Key,
hotel_name Varchar(100),
location Varchar(100),
price_per_night Decimal(10,2));
```

Hotels Table

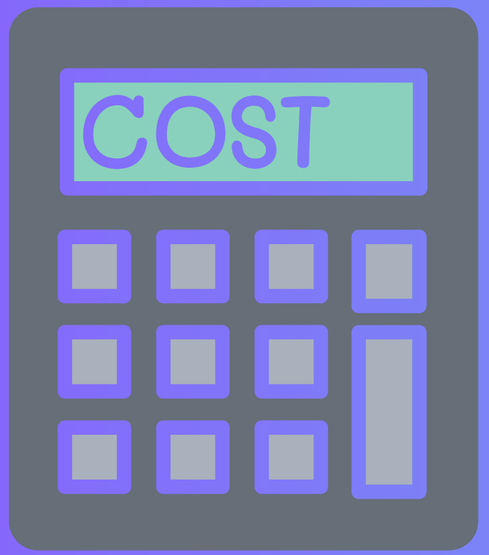
Field	Type	Null	Key	Default	Extra
hotel_id	int	NO	PRI	NULL	
hotel_name	varchar(100)	YES		NULL	
location	varchar(100)	YES		NULL	
price_per_night	decimal(10,2)	YES		NULL	

```
Create Table Activities (
activity_id Int Primary Key,
activity_name Varchar(100),
location Varchar(100),
Cost Decimal(10,2));
```

Activities Table

Field	Type	Null	Key	Default	Extra
activity_id	int	NO	PRI	NULL	
activity_name	varchar(100)	YES		NULL	
location	varchar(100)	YES		NULL	
cost	decimal(10,2)	YES		NULL	

Field	Type	Null	Key	Default	Extra
cost_id	int	NO	PRI	NULL	
package_id	int	YES	MUL	NULL	
hotel_id	int	YES	MUL	NULL	
activity_id	int	YES	MUL	NULL	
total_cost	decimal(10,2)	YES		NULL	



Costs Table

Create Table Costs (
cost_id int Primary Key,
package_id int,
hotel_id int,
activity_id int,
total_cost decimal(10,2),
Foreign Key (package_id) references
Packages(package_id),
Foreign Key (hotel_id) references Hotels(hotel_id),
Foreign Key (activity_id) references
Activities(activity_id),);

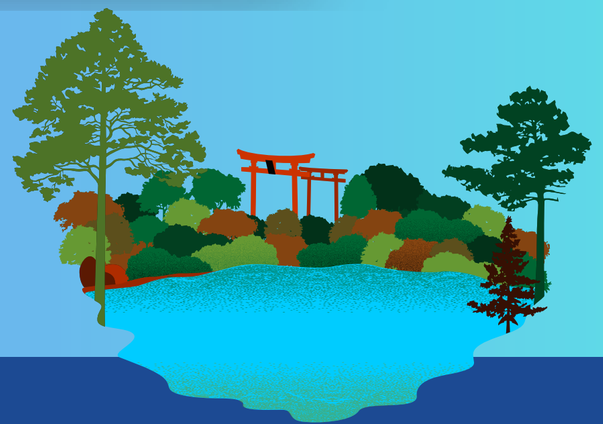
Step 2: Data Insertion

After setting up the schema, we now populate each table with initial sample data. This step ensures that we have enough variety to run real-time analytics and answer business queries later in the project.

Insert Into Users Table

```
Insert into Users (user_id, User_name, email) values
(1, 'Minnie Tanwar','minnie.tanwar@gmail.com'),
(2, 'Megha Bapat', 'bapat.megha@gmail.com'),
(3, 'Samiksha Singh','samiksha.singh@gmail.com'),
(4, 'Vedanshi Singh','veda.singh@gmail.com'),
(5, 'Rashmi','maam.rashmi@gmail.com'),
(6, 'Priya Jadeja','priya.jadega@gmail.com'),
(7, 'Shruti Tare','shruti.tare@gmail.com'),
(8, 'Komal Sharma','komal.sharma@yahoo.com'),
(9, 'Aditya Rao', 'aditya.rao@gmail.com'),
(10, 'Tanvi Desai','tanvi.desai@gmail.com'),
(11, 'Nikhil Singh','nikhil.singh@gmail.com'),
(12, 'Ms Tanwar', 'ms.tanwar@yahoo.com'),
(13, 'Devanshi Bapat','bapat.devu@gmail.com'),
(14, 'Neha Yadav','neha.Yadav@yahoo.com'),
(15, 'Aniket Reddy','aniket.reddy@gmail.com');
Select * from Users;
```

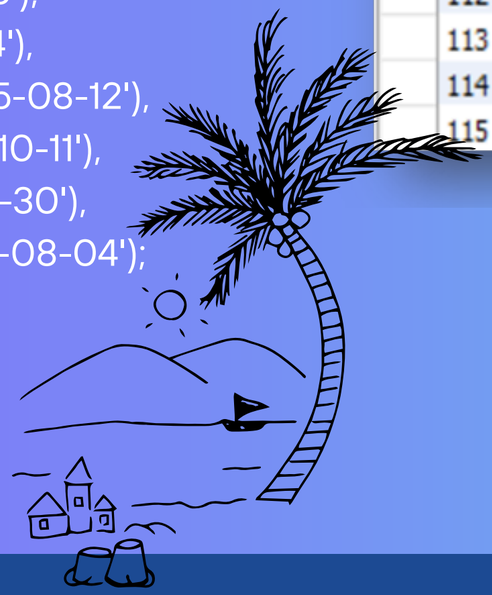
user_id	User_name	email
1	Minnie Tanwar	minnie.tanwar@gmail.com
2	Megha Bapat	bapat.megha@gmail.com
3	Samiksha Singh	samiksha.singh@gmail.com
4	Vedanshi Singh	veda.singh@gmail.com
5	Rashmi	maam.rashmi@gmail.com
6	Priya Jadeja	priya.jadega@gmail.com
7	Shruti Tare	shruti.tare@gmail.com
8	Komal Sharma	komal.sharma@yahoo.com
9	Aditya Rao	aditya.rao@gmail.com
10	Tanvi Desai	tanvi.desai@gmail.com
11	Nikhil Singh	nikhil.singh@gmail.com
12	Ms Tanwar	ms.tanwar@yahoo.com
13	Devanshi Bapat	bapat.devu@gmail.com
14	Neha Yadav	neha.Yadav@yahoo.com
15	Aniket Reddy	aniket.reddy@gmail.com
NULL	NULL	NULL

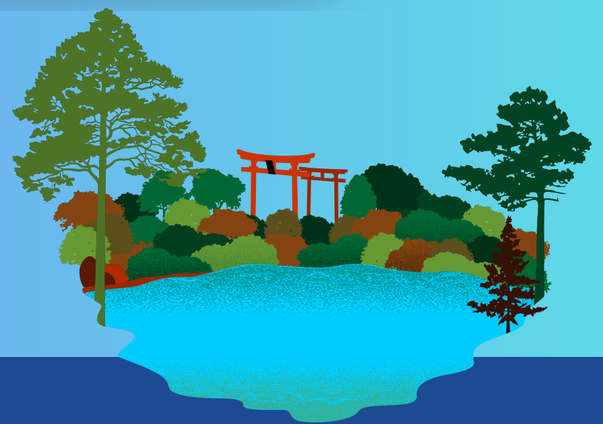


Insert Into Packages Table

```
Insert into Packages(package_id, user_id, destination, start_date, end_date) VALUES
'2025-07-20'),
(102, 2, 'Japan', '2025-08-05', '2025-08-15'),
(103, 3, 'Australia', '2025-09-01', '2025-09-12'),
(104, 4, 'Canada', '2025-07-22', '2025-08-02'),
(105, 5, 'Italy', '2025-10-10', '2025-10-20'),
(106, 6, 'Germany', '2025-07-01', '2025-07-11'),
(107, 7, 'Spain', '2025-08-18', '2025-08-28'),
(108, 8, 'United States', '2025-09-10', '2025-09-25'),
(109, 9, 'Switzerland', '2025-11-01', '2025-11-11'),
(110, 10, 'Greece', '2025-12-05', '2025-12-15'),
(111, 11, 'Norway', '2025-07-14', '2025-07-24'),
(112, 12, 'New Zealand', '2025-08-02', '2025-08-12'),
(113, 13, 'South Korea', '2025-10-01', '2025-10-11'),
(114, 14, 'Portugal', '2025-09-20', '2025-09-30'),
(115, 15, 'Netherlands', '2025-07-25', '2025-08-04');
select * from Packages;
```

package_id	user_id	destination	start_date	end_date
101	1	France	2025-07-10	2025-07-20
102	2	Japan	2025-08-05	2025-08-15
103	3	Australia	2025-09-01	2025-09-12
104	4	Canada	2025-07-22	2025-08-02
105	5	Italy	2025-10-10	2025-10-20
106	6	Germany	2025-07-01	2025-07-11
107	7	Spain	2025-08-18	2025-08-28
108	8	United States	2025-09-10	2025-09-25
109	9	Switzerland	2025-11-01	2025-11-11
110	10	Greece	2025-12-05	2025-12-15
111	11	Norway	2025-07-14	2025-07-24
112	12	New Zealand	2025-08-02	2025-08-12
113	13	South Korea	2025-10-01	2025-10-11
114	14	Portugal	2025-09-20	2025-09-30
115	15	Netherlands	2025-07-25	2025-08-04

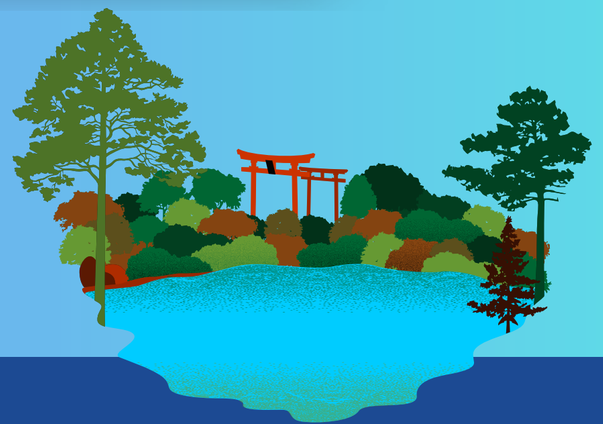




Insert Into Hotels Table

```
Insert into Hotels (hotel_id, hotel_name, location, price_per_night)
values (201, 'Hotel Le Paris', 'France', 225.00),
(202, 'Tokyo Garden Hotel', 'Japan', 130.00),
(203, 'Sydney Harbour Hotel', 'Australia', 160.00),
(204, 'Maple Leaf Suites', 'Canada', 170.00),
(205, 'Rome Palace Hotel', 'Italy', 140.00),
(206, 'Munich Comfort Hotel', 'Germany', 150.00),
(207, 'Madrid Grand Palace', 'Spain', 130.00),
(208, 'New York Elite Suites', 'United States', 200.00),
(209, 'Alpine Retreat', 'Switzerland', 210.00),
(210,'Santorini Sea Breeze', 'Greece', 125.00),
(211, 'Fjord View Hotel', 'Norway', 175.00),
(212, 'Queenstown Lake Resort', 'New Zealand', 165.00),
(213, 'Seoul City Inn', 'South Korea', 140.00),
(214, 'Porto Charm Stay', 'Portugal', 130.00),
(215, 'Amsterdam Canal Inn', 'Netherlands', 150.00);
select * from Hotels;
```

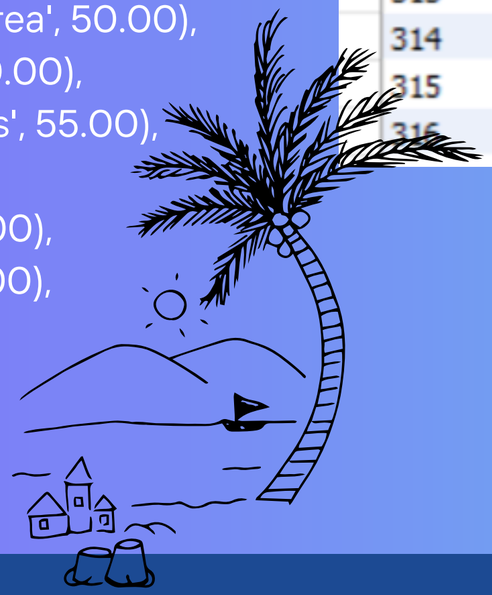
hotel_id	hotel_name	location	price_per_night
201	Hotel Le Paris	France	180.00
202	Tokyo Garden Hotel	Japan	130.00
203	Sydney Harbour Hotel	Australia	160.00
204	Maple Leaf Suites	Canada	170.00
205	Rome Palace Hotel	Italy	140.00
206	Munich Comfort Hotel	Germany	150.00
207	Madrid Grand Palace	Spain	130.00
208	New York Elite Suites	United States	200.00
209	Alpine Retreat	Switzerland	210.00
210	Santorini Sea Breeze	Greece	125.00
211	Fjord View Hotel	Norway	175.00
212	Queenstown Lake Re...	New Zealand	165.00
213	Seoul City Inn	South Korea	140.00
214	Porto Charm Stay	Portugal	130.00
215	Amsterdam Canal Inn	Netherlands	150.00

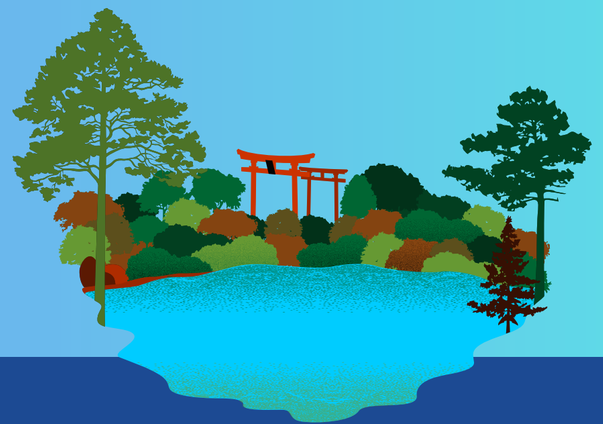


Insert Into Activities Table

Insert into Hotels (hotel_id, hotel_name, location, price_per_night)
values (301, 'Eiffel Tower Tour', 'France', 75.00),
(302, 'Mount Fuji Day Hike', 'Japan', 60.00),
(303, 'Sydney Opera House Visit', 'Australia', 55.00),
(304, 'Niagara Falls Cruise', 'Canada', 80.00),
(305, 'Colosseum Guided Tour', 'Italy', 70.00),
(306, 'Berlin Wall Museum', 'Germany', 50.00),
(307, 'Flamenco Show in Madrid', 'Spain', 45.00),
(308, 'Statue of Liberty Ferry Ride', 'United States', 65.00),
(309, 'Jungfrau Mountain Excursion', 'Switzerland', 95.00),
(310, 'Santorini Sunset Cruise', 'Greece', 85.00),
(311, 'Northern Lights Safari', 'Norway', 110.00),
(312, 'Milford Sound Kayaking', 'New Zealand', 90.00),
(313, 'Gyeongbokgung Palace Tour', 'South Korea', 50.00),
(314, 'Douro Valley Wine Tasting', 'Portugal', 60.00),
(315, 'Canal Cruise in Amsterdam', 'Netherlands', 55.00),
(316, 'Louvre Museum Visit', 'France', 65.00),
(317, 'Tokyo Disneyland Day Pass', 'Japan', 75.00),
(318, 'Blue Mountains Day Trip', 'Australia', 70.00),
(319, 'CN Tower EdgeWalk', 'Canada', 90.00),
(320, 'Venice Gondola Ride', 'Italy', 80.00),
.....

activity_id	activity_name	location	cost
301	Eiffel Tower Tour	France	75.00
302	Mount Fuji Day Hike	Japan	60.00
303	Sydney Opera House Visit	Australia	55.00
304	Niagara Falls Cruise	Canada	80.00
305	Colosseum Guided Tour	Italy	70.00
306	Berlin Wall Museum	Germany	50.00
307	Flamenco Show in Madrid	Spain	45.00
308	Statue of Liberty Ferry Ride	United States	65.00
309	Jungfrau Mountain Excursion	Switzerland	95.00
310	Santorini Sunset Cruise	Greece	85.00
311	Northern Lights Safari	Norway	110.00
312	Milford Sound Kayaking	New Zealand	90.00
313	Gyeongbokgung Palace Tour	South Korea	50.00
314	Douro Valley Wine Tasting	Portugal	60.00
315	Canal Cruise in Amsterdam	Netherlands	55.00
316	Louvre Museum Visit	France	65.00

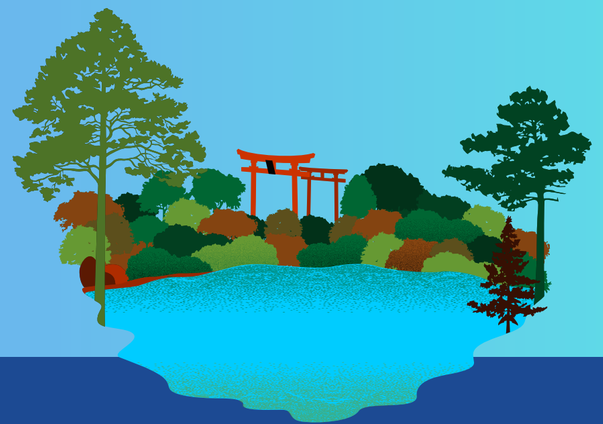




Insert Into Costs Table

```
Insert into Costs(cost_id, package_id, hotel_id, activity_id, total_cost)
values (401, 101, 201, 301, 1200.00),      -- (France)    225*5+75 = 975
(402, 101, 201, 316, 950.00),      -- (France)    175*5+65
(403, 102, 202, 302, 710.00),      -- (Japan )130*5+60 = 710
(404, 102, 202, 317, 935.00),      -- (Japan)     170*5+85
(405, 103, 203, 303, 855.00),      -- (Australia)  160*5+55 = 855
(406, 103, 203, 318, 910.00),      -- (Australia)  168*5 + 70
(407, 104, 204, 304, 930.00),      -- (Canada)    170*5+80 = 930
(408, 104, 204, 319, 1040.00),     -- (Canada)    190*5 + 90
(409, 105, 205, 305, 770.00),      -- (Italy)     140*5+70 = 770
(410, 105, 205, 320, 1000.00),     -- (Italy)     184*5 + 80
(411, 106, 206, 306, 800.00),      -- (Germany)   150*5+50 = 800
(412, 106, 206, 321, 840.00),      -- (Germany)   160*5 + 40
(413, 107, 207, 307, 695.00),      -- (Spain)     130*5+45 = 695
(414, 107, 207, 322, 900.00),      -- (Spain)     170*5 + 50
(415, 108, 208, 308, 1065.00),     -- (USA)       200*5+65 = 1065
(416, 108, 208, 323, 1300.00),     -- (USA)       210*5 + 250
(417, 109, 209, 309, 1145.00),     -- (Switzerland) 210*5+95 = 1145
(418, 109, 209, 324, 1100.00),     -- (Switzerland) 200*5 + 100
(419, 110, 210, 310, 710.00),      -- (Greece)    125*5+85 = 710
(420, 110, 210, 325, 960.00),      -- (Greece)    180*5 + 60
```

cost_id	package_id	hotel_id	activity_id	total_co
401	101	201	301	975.00
402	101	201	316	950.00
403	102	202	302	710.00
404	102	202	317	935.00
405	103	203	303	855.00
406	103	203	318	910.00
407	104	204	304	930.00
408	104	204	319	1040.00
409	105	205	305	770.00
410	105	205	320	1000.00
411	106	206	306	800.00
412	106	206	321	840.00
413	107	207	307	695.00
414	107	207	322	900.00
415	108	208	308	1065.00
416	108	208	323	1300.00



SQL Queries

Queries are instructions written in the SQL (Structured Query Language) that are used to interact with a database.

1. DDL - Data Definition Language

DDL stands for Data Definition Language.
It includes the SQL commands used to define or change the structure of a database and its tables.

Field	Type	Null	Key	Default	Extra
user_id	int	NO	PRI	NULL	
full_name	varchar(100)	YES		NULL	
email	varchar(100)	YES	UNI	NULL	
phone_number	varchar(15)	YES		NULL	

Field	Type	Null	Key	Default	Ext
hotel_id	int	NO	PRI	NULL	
hotel_name	varchar(100)	YES		NULL	
location	varchar(100)	YES		NULL	

--->ALTER TABLE: Add a column
ALTER TABLE Users ADD phone_number VARCHAR(15);

--->MODIFY – Change data type
ALTER TABLE Hotels MODIFY price_per_night DECIMAL(12,2);

---> CHANGE – Rename column and change type
ALTER TABLE Users CHANGE User_name full_name VARCHAR(100);

2. DML – Data Manipulation Language

DML stands for Data Manipulation Language.

It includes SQL commands used to manage and manipulate the actual data inside tables.

user_id	full_name	email	phone_number
1	Minnie Tanwar	minnie.tanwar@gmail.com	9876543210
2	Megha Bapat	bapat.megha@gmail.com	9876543211
3	Samiksha Singh	samiksha.singh@gmail.com	9876543213
4	Vedanshi Singh	veda.singh@gmail.com	9876543214
5	Rashmi	maam.rashmi@gmail.com	9876543215
6	Priya Jadeja	priya.jadega@gmail.com	9876543216
7	Shruti Tare	shruti.tare@gmail.com	9876543217
8	Komal Sharma	komal.sharma@yahoo.com	9876543218
9	Aditya Rao	aditya.rao@gmail.com	9876543219
10	Tanvi Desai	tanvi.desai@gmail.com	9876543220
11	Nikhil Singh	nikhil.singh@gmail.com	NULL
12	Ms Tanwar	ms.tanwar@yahoo.com	NULL
13	Devanshi Bapat	bapat.devu@gmail.com	NULL
14	Neha Yadav	neha.Yadav@yahoo.com	NULL
15	Aniket Reddy	aniket.reddy@gmail.com	NULL
16	Riya Kapoor	riya.kapoor@gmail.com	NULL

user_id	full_name	email	phone_number
1	Minnie Tanwar	minnie.tanwar@gmail.com	9876543210
2	Megha Bapat	bapat.megha@gmail.com	9876543211
3	Samiksha Singh	samiksha.singh@gmail.com	9876543213
4	Vedanshi Singh	veda.singh@gmail.com	9876543214
5	Rashmi	maam.rashmi@gmail.com	9876543215
6	Priya Jadeja	priya.jadega@gmail.com	9876543216
7	Shruti Tare	shruti.tare@gmail.com	9876543217
8	Komal Sharma	komal.sharma@yahoo.com	9876543218
9	Aditya Rao	aditya.rao@gmail.com	9876543219
10	Tanvi Desai	tanvi.desai@gmail.com	9876543220
11	Nikhil Singh	nikhil.singh@gmail.com	NULL
12	Ms Tanwar	ms.tanwar@yahoo.com	NULL
13	Devanshi Bapat	bapat.devu@gmail.com	NULL
14	Neha Yadav	neha.Yadav@yahoo.com	NULL
15	Aniket Reddy	aniket.reddy@gmail.com	NULL
NULL	NULL	NULL	NULL

Shruti Tare

--->INSERT INTO
INSERT INTO Users (user_id, full_name, email) VALUES (16, 'Riya Kapoor', 'riya.kapoor@gmail.com');

--->UPDATE
UPDATE Users SET phone_number = '9876543210' WHERE user_id = 1;

--->DELETE
DELETE FROM Users WHERE user_id = 16;

3. DQL – Data Query Language

DQL is used to ask questions to the database and get the data you want — without changing anything.

	full_name	email
▶	Minnie Tanwar	minnie.tanwar@gmail.com
	Megha Bapat	bapat.megha@gmail.com
	Samiksha Singh	samiksha.singh@gmail.com
	Vedanshi Singh	veda.singh@gmail.com
	Rashmi	maam.rashmi@gmail.com

---> SELECT + WHERE

SELECT full_name, email FROM Users WHERE user_id <= 5;

	TotalHotelCost
▶	1125.00

	cost_id	package_id	hotel_id	activity_id	total_cost
▶	401	101	201	301	1200.00
	408	104	204	319	1040.00
	415	108	208	308	1065.00
	416	108	208	323	1300.00
	417	109	209	309	1145.00
	418	109	209	324	1100.00
	422	111	211	326	1085.00
	424	112	212	327	1010.00
*	NULL	NULL	NULL	NULL	NULL

	package_id	user_id	destination	start_date	end_date
▶	101	1	France	2025-07-10	2025-07-20
*	NULL	NULL	NULL	NULL	NULL

4.Operators

SQL Operators
--->Arithmetic
SELECT price_per_night * 5 AS TotalHotelCost FROM Hotels WHERE hotel_id = 201;

---> Comparison
SELECT * FROM Costs WHERE total_cost > 1000;

---> Logical
SELECT * FROM Packages WHERE destination = 'France' AND user_id = 1;

user_id	full_name	email	phone_number
3	Samiksha Singh	samiksha.singh@gmail.com	9876543213
7	Shruti Tare	shruti.tare@gmail.com	9876543217
NULL	NULL	NULL	NULL

cost_id	package_id	hotel_id	activity_id	total_cost
416	108	208	323	1300.00
401	101	201	301	1200.00
417	109	209	309	1145.00
418	109	209	324	1100.00
422	111	211	326	1085.00
415	108	208	308	1065.00
408	104	204	319	1040.00
424	112	212	327	1010.00
410	105	205	320	1000.00
421	111	211	311	985.00
420	110	210	325	960.00
426	113	213	328	960.00
402	101	201	316	950.00
404	102	202	317	935.00
407	104	204	304	930.00
423	112	212	312	915.00

5.Like Operators and Order By Clause

--->Like
SELECT * FROM Users WHERE full_name LIKE 'S%';
Finds users whose name starts with 'S'.

--->Order BY
SELECT * FROM Costs ORDER BY total_cost DESC;
Displays packages from highest to lowest cost.

destination	total_packages
France	1
Japan	1
Australia	1
Canada	1
Italy	1
Germany	1
Spain	1
United States	1
Switzerland	1
Greece	1
Norway	1
New Zealand	1
South Korea	1
Portugal	1
Netherlands	1

6.Group By and Having Clause

--->Group by
SELECT destination, COUNT(*) AS total_packages FROM Packages GROUP BY destination;

---> Having
SELECT destination, COUNT(*) AS total_packages FROM Packages GROUP BY destination HAVING COUNT(*) >=1;

UPPER(full_name)
MINNIE TANWAR
MEGHA BAPAT
SAMIKSHA SINGH
VEDANSHI SINGH
RASHMI
PRIYA JADEJA
SHRUTI TARE
KOMAL SHARMA
ADITYA RAO
TANVI DESAI
NIKHIL SINGH
MS TANWAR
DEVANSHI BAPAT
NEHA YADAV
ANIKET REDDY

ROUND(total_cost, 0)
1200
950
710
935
855
910
930
1040
770
1000
800
840
695
900
1065

Duration
10
10
11
11
10
10
10
15
10
10
10
10
10
10
10

MAX(total_cost)
1300.00

package_id	user_id	destination	start_date	end_date
101	1	France	2025-07-10	2025-07-20
102	2	Japan	2025-08-05	2025-08-15
103	3	Australia	2025-09-01	2025-09-12
104	4	Canada	2025-07-22	2025-08-02
105	5	Italy	2025-10-10	2025-10-20
NULL	NULL	NULL	NULL	NULL

7.SQL Functions and Limit Clause

--->String
SELECT UPPER(full_name) FROM Users;
-->Math
SELECT ROUND(total_cost, 0) FROM Costs;;
-->Date
SELECT DATEDIFF(end_date, start_date) AS Duration FROM Packages;
-->Aggregate
SELECT MAX(total_cost) FROM Costs;

--->Limit Clause
SELECT * FROM Packages LIMIT 5;
Shows only first 5 packages

hotel_id	hotel_name	location	price_per_night
201	Hotel Le Paris	France	225.00
NULL	NULL	NULL	NULL

package_id	user_id	destination	start_date	end_date
109	9	Switzerland	2025-11-01	2025-11-11
115	15	Netherlands	2025-07-25	2025-08-04
113	13	South Korea	2025-10-01	2025-10-11
102	2	Japan	2025-08-05	2025-08-15
105	5	Italy	2025-10-10	2025-10-20
101	1	France	2025-07-10	2025-07-20
111	11	Norway	2025-07-14	2025-07-24
106	6	Germany	2025-07-01	2025-07-11
103	3	Australia	2025-09-01	2025-09-12
107	7	Spain	2025-08-18	2025-08-28
110	10	Greece	2025-12-05	2025-12-15
104	4	Canada	2025-07-22	2025-08-02
NULL	NULL	NULL	NULL	NULL

package_id	total_cost
101	1200.00
104	1040.00
108	1065.00
108	1300.00
109	1145.00
109	1100.00
111	1085.00
112	1010.00

8.Subqueries

--->Single-Row
SELECT * FROM Hotels WHERE price_per_night = (SELECT MAX(price_per_night) FROM Hotels);

--->Multi-Row
SELECT * FROM Packages WHERE user_id IN (SELECT user_id FROM Users WHERE email LIKE '%gmail.com');

--->Multi-Column
SELECT package_id, total_cost FROM Costs WHERE (hotel_id, activity_id) IN (SELECT hotel_id, activity_id FROM Costs WHERE total_cost > 1000);

full_name	destination
Minnie Tanwar	France
Megha Bapat	Japan
Samiksha Singh	Australia
Vedanshi Singh	Canada
Rashmi	Italy
Priya Jadeja	Germany
Shruti Tare	Spain
Komal Sharma	United States
Aditya Rao	Switzerland
Tanvi Desai	Greece
Nikhil Singh	Norway
Ms Tanwar	New Zealand
Devanshi Bapat	South Korea
Neha Yadav	Portugal
Aniket Reddy	Netherlands

destination	full_name
France	Minnie Tanwar
Japan	Megha Bapat
Australia	Samiksha Singh
Canada	Vedanshi Singh
Italy	Rashmi
Germany	Priya Jadeja
Spain	Shruti Tare
United States	Komal Sharma
Switzerland	Aditya Rao
Greece	Tanvi Desai
Norway	Nikhil Singh
New Zealand	Ms Tanwar
South Korea	Devanshi Bapat
Portugal	Neha Yadav
Netherlands	Aniket Reddy

user_id	full_name	email	phone_number	package_id	user_id	destination	start_date	end_date
1	Minnie Tanwar	minnie.tanwar@gmail.com	9876543210	101	1	France	2025-07-10	2025-07-20
2	Megha Bapat	bapat.megha@gmail.com	9876543211	102	2	Japan	2025-08-05	2025-08-15
3	Samiksha Singh	samiksha.singh@gmail.com	9876543213	103	3	Australia	2025-09-01	2025-09-12
4	Vedanshi Singh	veda	214	104	4	Canada	2025-07-22	2025-08-02
5	Rashmi	maam.rashmi@gmail.com	9876543215	105	5	Italy	2025-10-10	2025-10-20
6	Priya Jadeja	priya.jadega@gmail.com	9876543216	106	6	Germany	2025-07-01	2025-07-11
7	Shruti Tare	shruti.tare@gmail.com	9876543217	107	7	Spain	2025-08-18	2025-08-28
8	Komal Sharma	komal.sharma@yahoo.com	9876543218	108	8	United States	2025-09-10	2025-09-25
9	Aditya Rao	aditya.rao@gmail.com	9876543219	109	9	Switzerland	2025-11-01	2025-11-11
10	Tanvi Desai	tanvi.desai@gmail.com	9876543220	110	10	Greece	2025-12-05	2025-12-15
11	Nikhil Singh	nikhil.singh@gmail.com	NULL	111	11	Norway	2025-07-14	2025-07-24
12	Ms Tanwar	ms.tanwar@yahoo.com	NULL	112	12	New Zealand	2025-08-02	2025-08-12
13	Devanshi Bapat	bapat.devu@gmail.com	NULL	113	13	South Korea	2025-10-01	2025-10-11
14	Neha Yadav	neha.Yadav@yahoo.com	NULL	114	14	Portugal	2025-09-20	2025-09-30
15	Aniket Reddy	aniket.reddy@gmail.com	NULL	115	15	Netherlands	2025-07-25	2025-08-04

9.JOINS

--->Inner Join

```
SELECT U.full_name, P.destination FROM Users
U JOIN Packages P ON U.user_id = P.user_id;
```

--->Left Join

```
SELECT U.full_name, P.destination FROM Users
U LEFT JOIN Packages P ON U.user_id =
P.user_id;
```

--->Right Join

```
SELECT P.destination, U.full_name FROM
Packages P RIGHT JOIN Users U ON P.user_id
= U.user_id;
```

--->Full Outer Join

```
SELECT * FROM Users LEFT JOIN Packages
ON Users.user_id = Packages.user_id
UNION
SELECT * FROM Users RIGHT JOIN Packages ON
Users.user_id = Packages.user_id;
```

package_id	full_name	destination	total_cost
101	Minnie Tanwar	France	1200.00
104	Vedanshi Singh	Canada	1040.00
108	Komal Sharma	United States	1065.00
108	Komal Sharma	United States	1300.00
109	Aditya Rao	Switz	.00
109	Aditya Rao	Switzerland	1100.00
111	Nikhil Singh	Norway	1085.00
112	Ms Tanwar	New Zealand	1010.00

United States

10.Views

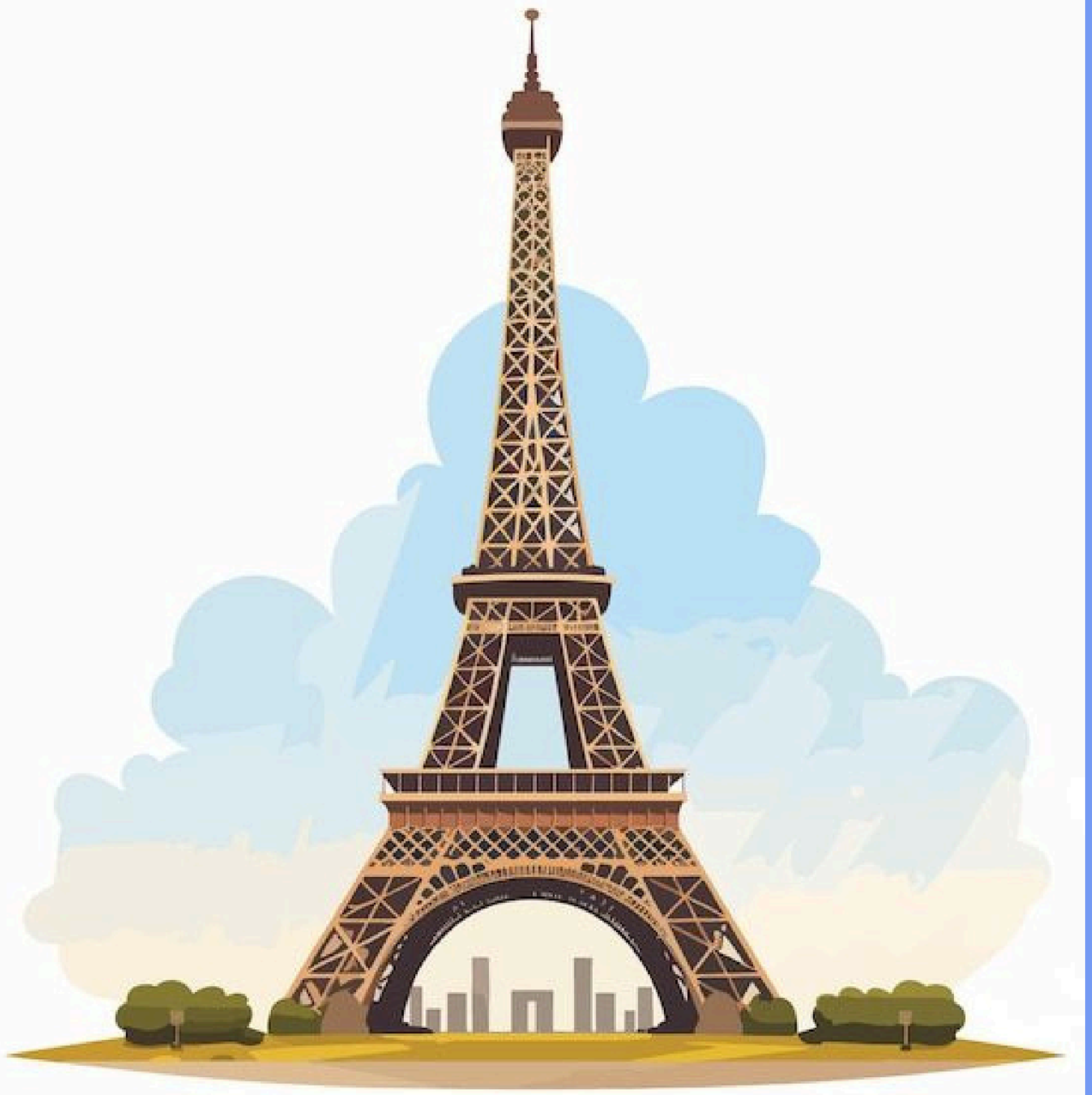
```
CREATE VIEW PackageCostsView AS
SELECT P.package_id, U.full_name,
P.destination, C.total_cost FROM
Packages P
JOIN Users U ON P.user_id = U.user_id
JOIN Costs C ON P.package_id =
C.package_id;
```

-->Creates a virtual table showing user, package, and cost info.

```
SELECT * FROM PackageCostsView
WHERE total_cost > 1000;
```

SQL Queries to Explore Insights

Now that the database is set up with records, we'll begin answering real-world questions through SQL queries. These queries will help:



Which users have booked trip to ‘France’?

```
SELECT u.user_id, u.Full_name,
p.destination FROM Users u
JOIN Packages p ON u.user_id =
p.user_id
WHERE p.destination = 'France';
```

	user_id	full_name	destination
▶	1	Minnie Tanwar	France

	user_id	Full_name	total_spent
▶	8	Komal Sharma	2365.00
	9	Aditya Rao	2245.00
	1	Minnie Tanwar	2150.00
	11	Nikhil Singh	2070.00
	4	Vedanshi Singh	1970.00
	12	Ms Tanwar	1925.00
	5	Rashmi	1770.00
	3	Samiksha Singh	1765.00
	13	Devanshi Bapat	1710.00
	15	Aniket Reddy	1675.00
	10	Tanvi Desai	1670.00
	2	Megha Bapat	1645.00
	6	Priya Jadeja	1640.00
	7	Shruti Tare	1595.00
	14	Neha Yadav	1535.00

What is the total cost for each user across all their packages?

```
SELECT u.user_id, u.full_name,
SUM(c.total_cost) AS total_spent
FROM Users u
JOIN Packages p ON u.user_id = p.user_id
JOIN Costs c ON p.package_id =
c.package_id
GROUP BY u.user_id, u.full_name
ORDER BY total_spent DESC;
```

	destination	avg_price
▶	France	225.000000
	Switzerland	210.000000
	United States	200.000000
	Norway	175.000000
	Canada	170.000000
	New Zealand	165.000000
	Australia	160.000000
	Germany	150.000000
	Netherlands	150.000000
	Italy	140.000000
	South Korea	140.000000
	Japan	130.000000
	Spain	130.000000
	Portugal	130.000000
	Greece	125.000000

Show average hotel Price per destination.

```
SELECT h.location AS destination,
AVG(h.price_per_night) AS avg_price
FROM Hotels h
GROUP BY h.location
ORDER BY avg_price DESC;
```

	full_name	destination	max_spend
▶	Komal Sharma	United States	1300.00

Which user spent the most in a single package?

```
SELECT u.Full_name, p.destination,
MAX(c.total_cost) AS max_spend
FROM Users u
JOIN Packages p ON u.user_id = p.user_id
JOIN Costs c ON p.package_id = c.package_id
GROUP BY u.Full_name, p.destination
ORDER BY max_spend DESC
LIMIT 1;
```

package_id	user_id	destination	start_date	end_date
101	1	France	2025-07-10	2025-07-20
104	4	Canada	2025-07-22	2025-08-02
106	6	Germany	2025-07-01	2025-07-11
111	11	Norway	2025-07-14	2025-07-24
115	15	Netherlands	2025-07-25	2025-08-04
NULL	NULL	NULL	NULL	NULL

Which packages are starting in the next 30 days?

```
SELECT *
FROM Packages
WHERE start_date BETWEEN CURDATE()
AND DATE_ADD(CURDATE(), INTERVAL 30
DAY);
```

destination	total_cost
France	1200.00
Canada	1040.00
United States	1065.00
United States	1300.00
Switzerland	1145.00
Switzerland	1100.00
Norway	1085.00
New Zealand	1010.00

Find destinations where total cost exceeds $1000?

```
SELECT DISTINCT p.destination, c.total_cost
FROM Packages p
JOIN Costs c ON p.package_id =
c.package_id
WHERE c.total_cost > 1000;
```

	destination	package_count
▶	France	1
	Japan	1
	Australia	1
	Canada	1
	Italy	1
	Germany	1
	Spain	1
	United States	1
	Switzerland	1
	Greece	1
	Norway	1
	New Zealand	1
	South Korea	1
	Portugal	1
	Netherlands	1

Count of package per destination (popularity rank)

```
SELECT destination, COUNT(*) AS
package_count
FROM Packages
GROUP BY destination
ORDER BY package_count DESC;
```

	hotel_name	activity_name	total_cost
▶	Tokyo Garden Hotel	Mount Fuji Day Hike	710.00
	Rome Palace Hotel	Colosseum Guided Tour	
	Madrid Grand Palace	Flamenco Show in Madrid	
	Santorini Sea Breeze	Santorini Sunset Cruise	710.00
	Seoul City Inn	Gyeongbokgung Palace Tour	750.00
	Porto Charm Stay	Douro Valley Wine Tasting	710.00

List all hotel-activity combinations with total cost < ₹800.

```
SELECT h.hotel_name, a.activity_name,
c.total_cost
FROM Costs c
JOIN Hotels h ON c.hotel_id = h.hotel_id
JOIN Activities a ON c.activity_id =
a.activity_id
WHERE c.total_cost < 800;
```

location	avg_activity_cost
United States	157.500000
Norway	102.500000
Switzerland	97.500000
New Zealand	87.500000
Canada	85.000000
Italy	75.000000
Greece	72.500000
France	70.000000
Japan	67.500000
Australia	62.500000
South Korea	55.000000
Netherlands	50.000000
Spain	47.500000
Germany	45.000000
Portugal	42.500000

Show all activity names and their average cost per location.

```
SELECT location, AVG(cost) AS
avg_activity_cost
FROM Activities
GROUP BY location
ORDER BY avg_activity_cost DESC;
```

user_id	full_name	num_packages
1	Minnie Tanwar	1
2	Megha Bapat	1
3	Samiksha Singh	1
4	Vedanshi Singh	1
5	Rashmi	1
6	Priya Jadeja	1
7	Shruti Tare	1
8	Komal Sharma	1
9	Aditya Rao	1
10	Tanvi Desai	1
11	Nikhil Singh	1
12	Ms Tanwar	1
13	Devanshi Bapat	1
14	Neha Yadav	1
15	Aniket Reddy	1

How many users booked more than one package?

```
SELECT u.user_id, u.Full_name,
COUNT(p.package_id) AS num_packages
FROM Users u
JOIN Packages p ON u.Full_id = p.user_id
GROUP BY u.user_id, u.Full_name
HAVING COUNT(p.package_id) > 1;
```

SUMMERY

Build a Tour Package Planner with Price Estimator that:

Allows users to design and customize their own travel packages.

Connects selected packages to hotels and activities across various destinations.

Calculates total trip costs based on hotel stays, activity choices, and duration.

Provides real-time cost estimates and a detailed breakdown for better planning.

-->This project covers:

-->Real-world data design

-->Full SQL concept usage (DDL, DML, DQL)

-->Joins and subqueries

-->Functions and views

-->Estimations using math on cost