University of Dhaka

Department of Computer Science and Engineering

CSE2211: Database Management Systems-1

Lab Project

VoyageVista: Travel Booking Platform

A Comprehensive Database Management System

Lab Group: A (ODD)

H.M. Mehedi Hasan (Roll: 13)
MD. Abu Bakar Siddique (Roll: 47)

Submitted To:

Mr. Abu Ahmed Ferdaus, Dept. of CSE, University of Dhaka
Mr. Redwan Ahmed Rizvee, Dept of CSE, University of Dhaka

Submission Date: July 17, 2025

Contents

1 Introduction

1.1 Project Overview

VoyageVista represents a modern approach to **travel booking systems**, combining sophisticated database design with contemporary web technologies. The platform addresses the growing need for efficient travel management systems that can handle complex relationships between **users, travel packages, destinations, bookings, and promotional offers**.

1.2 System Objectives

The primary objectives of the VoyageVista system include:

- Efficient management of travel packages and destinations
- Seamless user registration and booking processes
- Comprehensive administrative controls and analytics
- Secure authentication and authorization mechanisms
- Scalable database architecture supporting concurrent operations
- Implementation of advanced database concepts and query optimization

1.3 Technology Stack

The system utilizes a modern technology stack:

- **Database:** PostgreSQL 15 with UUID primary keys
- **Backend:** FastAPI with SQLModel and Alembic migrations
- **Frontend:** Next.js 15 with React 19 and TypeScript
- **Caching:** Redis for session management
- **Storage:** Supabase for file management
- **Authentication:** JWT with dual role system

2 Database Design

2.1 Database Schema Overview

The VoyageVista database consists of **9 main entities** designed to handle the complete travel booking ecosystem. The schema incorporates various data types, constraints, and relationships to ensure data integrity and efficient operations.

2.2 Schema Diagram

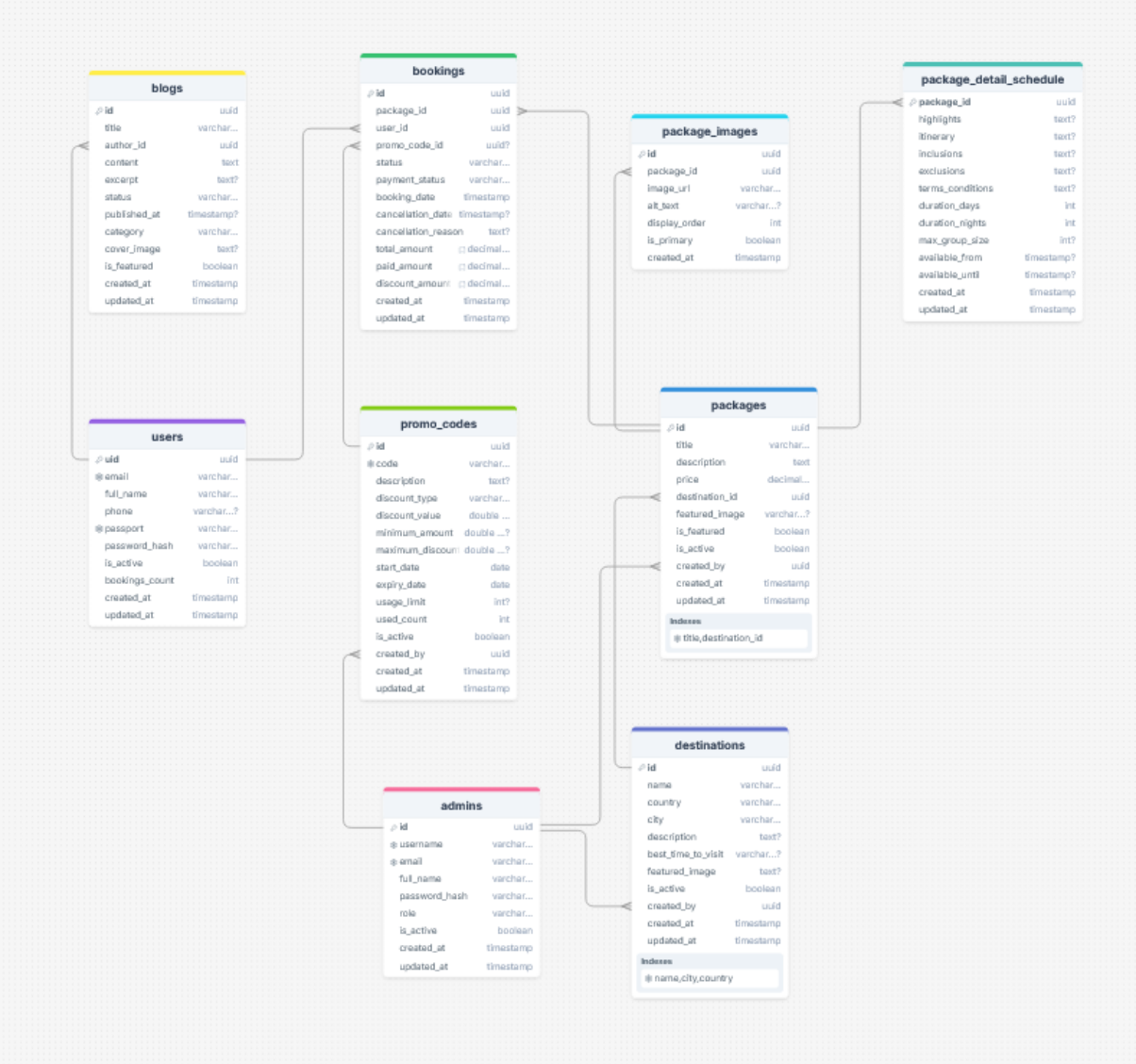

Figure 1: Schema Diagram

2.3 Entity-Relationship Diagram

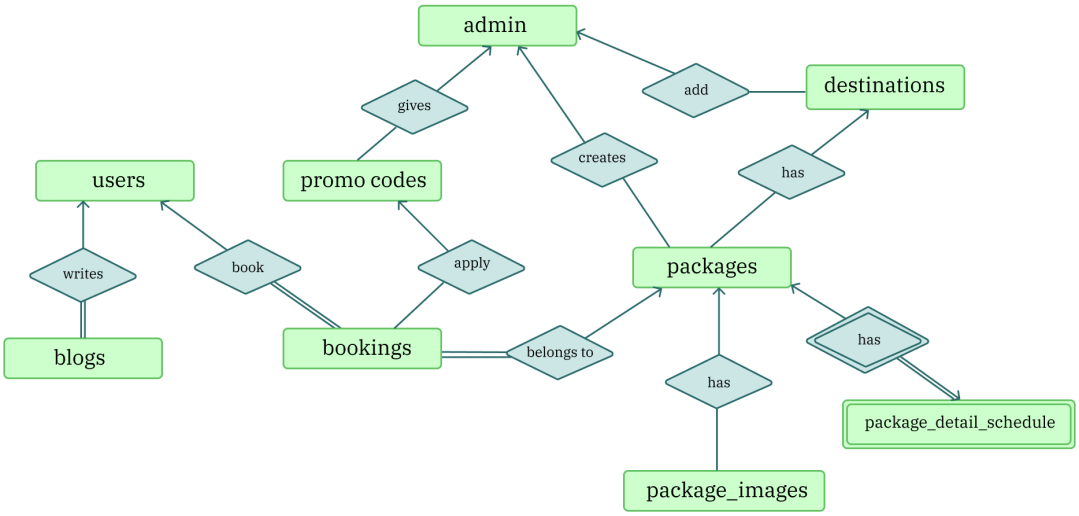

Figure 2: ER Diagram

2.4 Entity Descriptions

2.4.1 Users Table

Stores customer information with comprehensive profile data including authentication credentials, contact information, and passport details for international travel verification.

2.4.2 Admins Table

Manages administrative users with role-based access control, supporting different administrative levels and secure authentication mechanisms.

2.4.3 Destinations Table

Contains travel destination information including geographical data, descriptions, and metadata for package association and user browsing.

2.4.4 Packages Table

Central entity for travel packages, linking destinations with pricing, availability, and promotional features.

2.4.5 Package Detail Schedule Table

Extended package information including detailed itineraries, inclusions, exclusions, and scheduling constraints.

2.4.6 Package Images Table

Image management for packages with support for multiple images, ordering, and primary image designation.

2.4.7 Bookings Table

Transaction records for user bookings including payment tracking, status management, and promotional code integration.

2.4.8 Promo Codes Table

Promotional discount system with flexible discount types, usage limits, and expiration management.

2.4.9 Blogs Table

Content management system for travel-related articles, guides, and promotional content with publishing workflow.

2.5 Database Constraints

2.5.1 Primary Key Constraints

All tables implement UUID primary keys for enhanced security and distributed system compatibility:

- `users.uid` - Primary key for users table
- `admins.id` - Primary key for admins table
- `destinations.id` - Primary key for destinations table
- `packages.id` - Primary key for packages table
- `bookings.id` - Primary key for bookings table
- `promo_codes.id` - Primary key for promo codes table
- `blogs.id` - Primary key for blogs table

2.5.2 Foreign Key Constraints

Referential integrity maintained through foreign key relationships:

- `packages.destination_id` → `destinations.id`
- `packages.created_by` → `admins.id`
- `destinations.created_by` → `admins.id`
- `bookings.user_id` → `users.uid`
- `bookings.package_id` → `packages.id`
- `bookings.promo_code_id` → `promo_codes.id`
- `blogs.author_id` → `users.uid`
- `promo_codes.created_by` → `admins.id`
- `package_detail_schedule.package_id` → `packages.id` (CASCADE DELETE)
- `package_images.package_id` → `packages.id` (CASCADE DELETE)

2.5.3 Unique Constraints

Data uniqueness enforced at database level:

- `users.email` - Unique user email identification
- `users.passport` - Unique passport numbers
- `admins.username` - Unique admin usernames
- `admins.email` - Unique admin emails
- `promo_codes.code` - Unique promotional codes
- `destinations.name, city, country` - Unique destination location combination
- `packages.title, destination_id` - Unique package title per destination

2.5.4 Check Constraints

Business logic enforcement through check constraints:

- `packages.price >= 0` - Non-negative pricing
- `promo_codes.discount_value >= 0` - Valid discount values
- `bookings.total_amount >= 0` - Non-negative amounts
- `package_detail_schedule.duration_days >= 1` - Minimum duration

3 Database Implementation

3.1 DDL Statements

3.1.1 Users Table Creation

```sql
CREATE TABLE users (
    uid uuid NOT NULL,
    email character varying(255) NOT NULL,
    full_name character varying(255) NOT NULL,
    phone character varying(20),
    passport character varying(255) NOT NULL,
    password_hash character varying(255) NOT NULL,
    is_active boolean NOT NULL,
    bookings_count integer NOT NULL,
    created_at timestamp without time zone NOT NULL,
    updated_at timestamp without time zone NOT NULL,

    CONSTRAINT unique_user_email UNIQUE (email),
    CONSTRAINT unique_user_passport UNIQUE (passport),
    CONSTRAINT users_pkey PRIMARY KEY (uid)
);

SELECT * FROM users;
```

Listing 1: Users Table DDL with Data Query

3.1.2 Admins Table Creation

```sql
CREATE TABLE admins (
    id uuid NOT NULL,
    username character varying(50) NOT NULL,
    email character varying(255) NOT NULL,
    full_name character varying(255) NOT NULL,
    password_hash character varying(255) NOT NULL,
    role character varying(50) NOT NULL,
    is_active boolean NOT NULL,
    created_at timestamp without time zone NOT NULL,
    updated_at timestamp without time zone NOT NULL,

    CONSTRAINT unique_admin_username UNIQUE (username),
    CONSTRAINT unique_admin_email UNIQUE (email),
    CONSTRAINT admins_pkey PRIMARY KEY (id)
);

SELECT * FROM admins;
```

Listing 2: Admins Table DDL with Data Query

3.1.3 Destinations Table Creation

```sql
CREATE TABLE destinations (
    id uuid NOT NULL,
    name character varying(255) NOT NULL,
```

```
    country character varying(100) NOT NULL,
    city character varying(100) NOT NULL,
    description text,
    best_time_to_visit character varying(100),
    featured_image text,
    is_active boolean NOT NULL,
    created_by uuid NOT NULL,
    created_at timestamp NOT NULL,
    updated_at timestamp NOT NULL,

    CONSTRAINT unique_destination_name_location UNIQUE (name, city,
   country),
    CONSTRAINT destinations_created_by_fkey FOREIGN KEY (created_by)
   REFERENCES admins(id),
    CONSTRAINT destinations_pkey PRIMARY KEY (id)
);

SELECT * FROM destinations;
```

Listing 3: Destinations Table DDL with Data Query

3.1.4 Packages Table Creation

```
CREATE TABLE packages (
    id uuid NOT NULL,
    title character varying(255) NOT NULL,
    description text NOT NULL,
    price numeric(10,2) NOT NULL,
    destination_id uuid NOT NULL,
    featured_image character varying(500),
    is_featured boolean NOT NULL,
    is_active boolean NOT NULL,
    created_by uuid NOT NULL,
    created_at timestamp NOT NULL,
    updated_at timestamp NOT NULL,

    CONSTRAINT packages_destination_id_fkey FOREIGN KEY (destination_id
   ) REFERENCES destinations(id),
    CONSTRAINT packages_created_by_fkey FOREIGN KEY (created_by)
   REFERENCES admins(id),
    CONSTRAINT unique_title_per_destination UNIQUE (title,
   destination_id),
    CONSTRAINT packages_pkey PRIMARY KEY (id)
);

SELECT * FROM packages;
```

Listing 4: Packages Table DDL with Data Query

3.1.5 Bookings Table Creation

```
CREATE TABLE bookings (
    id uuid NOT NULL,
    package_id uuid NOT NULL,
    user_id uuid NOT NULL,
    promo_code_id uuid,
```

```
    status character varying(20) NOT NULL,
    payment_status character varying(20) NOT NULL,
    booking_date timestamp without time zone NOT NULL,
    cancellation_date timestamp without time zone,
    cancellation_reason text,
    total_amount numeric(10,2) DEFAULT 0.0 NOT NULL,
    paid_amount numeric(10,2) DEFAULT 0.0 NOT NULL,
    discount_amount numeric(10,2) DEFAULT 0.0 NOT NULL,
    created_at timestamp without time zone NOT NULL,
    updated_at timestamp without time zone NOT NULL,

    CONSTRAINT bookings_package_id_fkey FOREIGN KEY (package_id)
   REFERENCES packages(id),
    CONSTRAINT bookings_user_id_fkey FOREIGN KEY (user_id) REFERENCES
   users(uid),
    CONSTRAINT bookings_promo_code_id_fkey FOREIGN KEY (promo_code_id)
   REFERENCES promo_codes(id),
    CONSTRAINT bookings_pkey PRIMARY KEY (id)
);

SELECT * FROM bookings;
```

Listing 5: Bookings Table DDL with Data Query

3.1.6 Blogs Table Creation

```
CREATE TABLE blogs (
    id uuid NOT NULL PRIMARY KEY,
    title character varying(255) NOT NULL,
    author_id uuid NOT NULL,
    content text NOT NULL,
    excerpt text,
    status character varying(20) NOT NULL,
    published_at timestamp without time zone,
    category character varying(50) NOT NULL,
    cover_image text,
    is_featured boolean NOT NULL,
    created_at timestamp without time zone NOT NULL,
    updated_at timestamp without time zone NOT NULL,

    CONSTRAINT blogs_author_id_fkey
    FOREIGN KEY (author_id) REFERENCES public.users(uid)
);
```

Listing 6: Blogs Table DDL

3.1.7 Promo Codes Table Creation

```
CREATE TABLE promo_codes (
    id uuid NOT NULL PRIMARY KEY,
    code character varying(50) NOT NULL,
    description text,
    discount_type character varying(20) NOT NULL,
    discount_value double precision NOT NULL,
    minimum_amount double precision,
    maximum_discount double precision,
```

```
    start_date date NOT NULL,
    expiry_date date NOT NULL,
    usage_limit integer,
    used_count integer NOT NULL,
    is_active boolean NOT NULL,
    created_by uuid NOT NULL,
    created_at timestamp NOT NULL,
    updated_at timestamp NOT NULL,

    CONSTRAINT promo_codes_code_key UNIQUE (code),
    CONSTRAINT promo_codes_created_by_fkey
    FOREIGN KEY (created_by) REFERENCES public.admins(id)
);
```

Listing 7: Promo Codes Table DDL

3.1.8 Package Detail Schedule Table Creation

```
CREATE TABLE package_detail_schedule (
    package_id uuid NOT NULL PRIMARY KEY,
    highlights text,
    itinerary text,
    inclusions text,
    exclusions text,
    terms_conditions text,
    duration_days integer NOT NULL,
    duration_nights integer NOT NULL,
    max_group_size integer,
    available_from timestamp without time zone,
    available_until timestamp without time zone,
    created_at timestamp without time zone NOT NULL,
    updated_at timestamp without time zone NOT NULL,

    CONSTRAINT package_detail_schedule_package_id_fkey
    FOREIGN KEY (package_id) REFERENCES public.packages(id) ON DELETE
   CASCADE
);
```

Listing 8: Package Detail Schedule Table DDL

3.1.9 Package Images Table Creation

```
CREATE TABLE package_images (
    id uuid NOT NULL PRIMARY KEY,
    package_id uuid NOT NULL,
    image_url character varying(500) NOT NULL,
    alt_text character varying(255),
    display_order integer NOT NULL,
    is_primary boolean NOT NULL,
    created_at timestamp without time zone NOT NULL,

    CONSTRAINT package_images_package_id_fkey
    FOREIGN KEY (package_id) REFERENCES public.packages(id) ON DELETE
   CASCADE
);
```

Listing 9: Package Images Table DDL

3.2 Populated Table Previews

3.2.1 admins Table Snapshot

	id [PK] uuid	username character varying (50)	email character varying (255)	full_name character varying (255)	password_hash character varying	role character varying (50)	is_active boolean	created_at timestamp without tim	updated_at timestamp without time zone
1	929b7ad6-d9c5-4cf8-b...	Mehedi26696	hasanmehedi26696@gmail.com	H.M.Mehedi Hasan	$2b$12$2SNe...	admin	true	2025-07-09 00:57:4...	2025-07-09 01:01:53.472271
2	3a24275f-89b4-49c5-8...	admin	admin@vistavoyage.com	System Administrator	$2b$12$N1D...	super_admin	true	2025-07-08 21:02:0...	2025-07-13 22:52:59.375718
3	c792ac34-72ed-4c04-9...	abs	bojackabs@gmail.com	ABS ABS	$2b$12$/oue...	admin	true	2025-07-14 01:02:0...	2025-07-14 01:02:07.005797
4	123e4567-e89b-12d3-...	migration_admin	migration_admin@example.com	Migration Admin	dummyhash	admin	true	2025-07-15 22:30:1...	2025-07-15 22:30:18.118188
5	a1a1a1a1-a1a1-a1a1-a...	admin_john	john@travel.com	John Admin	hashed_pass...	superadmin	true	2025-07-16 01:32:1...	2025-07-16 01:32:18.642621

3.2.2 users Table Snapshot

	uid [PK] uuid	email character varying (255)	full_name character varying (255)	phone character varying (20)	passport character varying (255)	password_hash character varying	is_active boolean	bookings_co integer	created_at timestamp wit	updated_at timestamp witho
1	5a85f0a3-6cb0-4578-b6ec-7...	hasanmehedi26696@gmail.com	Mehedi Hasan	+8801319926696	12345678	$2b$12$Chm...	true	0	2025-07-0...	2025-07-08 ...
2	144be6a9-8794-46d6-9a9c-...	jfwejrw@gmail.com	New	+8801319926696	2375983	$2b$12$Mwl...	true	0	2025-07-1...	2025-07-15 ...
3	c361a0ec-603e-482a-b52e-7...	bojackabs@gmail.com	Abu Bakar Siddique	+880131992657	5789273982	$2b$12$SLbv...	true	0	2025-07-1...	2025-07-16 ...
4	1f37c497-2b0a-4c20-b5f3-8...	alice@example.com	Alice Wonderland	01710000001	P12345601	hash_pw1	true	2	2025-07-1...	2025-07-16 ...
5	d72a453b-e1cc-4bd8-84ea-...	bob@example.com	Bob Builder	01710000002	P12345602	hash_pw2	true	1	2025-07-1...	2025-07-16 ...
6	5e99a3c8-19e2-4a90-8049-...	charlie@example.com	Charlie Chaplin	01710000003	P12345603	hash_pw3	true	0	2025-07-1...	2025-07-16 ...
7	f441d68e-0a31-4b20-93d1-7...	diana@example.com	Diana Prince	01710000004	P12345604	hash_pw4	true	3	2025-07-1...	2025-07-16 ...
8	cc63af5e-feb6-4873-b219-5...	ethan@example.com	Ethan Hunt	01710000005	P12345605	hash_pw5	true	5	2025-07-1...	2025-07-16 ...
9	b9dc1d92-d3aa-4db0-9880-...	fiona@example.com	Fiona Gallagher	01710000006	P12345606	hash_pw6	true	1	2025-07-1...	2025-07-16 ...

3.2.3 packages Table Snapshot

	id [PK] uuid	title character varying (2	description text	price numeric (10,2)	destination_id uuid	featured_image character varying (500)	is_featured boolean	is_active boolean	created_at timestamp w	updated_at timestamp w	created_ uuid
1	39f5d135-a1f3-434f-9ff...	Discover Sundar...	Explore the world's largest m...	4000.00	e9f85003-a946-4687-8811-8...	https://tywqqefmllgse...	false	true	2025-07-...	2025-07-...	3a2...
2	038a03ac-7383-4f96-bb...	Relaxing Beach ...	Enjoy the longest natural sea ...	1500.00	09e5b484-fdfb-466f-a0af-05...	https://tywqqefmllgse...	false	true	2025-07-...	2025-07-...	123...
3	b0b981cf-1859-4439-b6...	Mystic Sylhet Te...	Unwind in the serene landsca...	1500.00	7cce5e46-6227-455f-b4a3-f...	https://tywqqefmllgse...	false	true	2025-07-...	2025-07-...	123...
4	f3b2a1d1-9e0c-4b5d-9c...	Cox's Beach Bon...	3-day fun-filled tour to Cox's ...	12500.00	d1b5ea5a-12e9-4b52-8e3d-...	https://example.com/i...	true	true	2025-07-...	2025-07-...	a1a...
5	f3b2a1d1-9e0c-4b5d-9c...	Sajek Cloudy Es...	2-night stay in Sajek with sun...	10500.00	d1b5ea5a-12e9-4b52-8e3d-...	https://example.com/i...	true	true	2025-07-...	2025-07-...	a1a...
6	f3b2a1d1-9e0c-4b5d-9c...	Bandarban Trek...	4-day nature and hiking adve...	14500.00	d1b5ea5a-12e9-4b52-8e3d-...	https://example.com/i...	false	true	2025-07-...	2025-07-...	a1a...
7	f3b2a1d1-9e0c-4b5d-9c...	Saint Martin Get...	Island vibes and coral reef sn...	13500.00	d1b5ea5a-12e9-4b52-8e3d-...	https://example.com/i...	true	true	2025-07-...	2025-07-...	a1a...
8	f3b2a1d1-9e0c-4b5d-9c...	Jaflong Nature ...	Green hills, river boating, and ...	9800.00	d1b5ea5a-12e9-4b52-8e3d-...	https://example.com/i...	true	true	2025-07-...	2025-07-...	a1a...

3.2.4 package-detail-schedule Table Snapshot

	package_id uuid	highlights text	itinerary text	inclusions text	exclusions text	terms_conditions text	duration_d integer	duration_nigh integer	max_gro integer	available_from timestamp without tir	available_until timestamp without tim	created_at timestamp without ti	updated timestai
1	39f5d135-a1f3-4...	Boat ride, wildli...	Day 1: Arrival at K...	Accommodati...	Personal expen...	Bookings are non-r...	5	6	10	2025-07-15 00:00...	2025-07-24 00:00:...	2025-07-15 16:5...	202...
2	038a03ac-7383-...	Sunbathing, be...	Day 1: Arrival and ...	Accommodati...	Personal expen...	Full refund availabl...	5	5	5	2025-07-16 00:00...	2025-07-25 00:00:...	2025-07-14 13:4...	202...
3	b0b981cf-1859-4...	Tea garden tou...	Day 1: Arrival in S...	Accommodati...	Lunch/dinner, p...	Advance booking r...	5	5	10	2025-07-16 00:00...	2025-07-30 00:00:...	2025-07-15 18:1...	202...
4	f3b2a1d1-9e0c-4...	Beach BBQ, Se...	Day 1: Arrival\nDa...	Hotel, BBQ, Tr...	Lunch	No refund after 7 d...	3	2	30	2025-07-16 01:38...	2025-10-16 01:38:...	2025-07-16 01:3...	202...
5	f3b2a1d1-9e0c-4...	Sunrise trekkin...	Day 1: Travel\nDa...	Stay, Breakfast	Guide fee	ID required	3	2	20	2025-07-16 01:38...	2025-09-16 01:38:...	2025-07-16 01:3...	202...
6	f3b2a1d1-9e0c-4...	Waterfalls, hike	Day 1: Arrival\nDa...	Guide, Food, T...	Shoes	No smoking	4	3	15	2025-07-16 01:38...	2026-01-16 01:38:...	2025-07-16 01:3...	202...
7	f3b2a1d1-9e0c-4...	Boat ride, snor...	Day 1: Ferry\nDay...	Resort, Meals	Alcohol	Wear safety gear	3	2	25	2025-07-16 01:38...	2025-11-16 01:38:...	2025-07-16 01:3...	202...
8	f3b2a1d1-9e0c-4...	River boating, t...	Day 1: Travel\nDa...	Hotel, Breakfast	Lunch	Family-friendly only	3	2	40	2025-07-16 01:38...	2025-12-16 01:38:...	2025-07-16 01:3...	202...

3.2.5 package-images Table Snapshot

	id [PK] uuid	package_id uuid	image_url character varying (500)	alt_text character varying (255)	display_order integer	is_primary boolean	created_at timestamp without time zone
1	403bcf8b-d674-4cdf-81bc-f07524b32b9d	39f5d135-a1f3-434f-9ff8-ce34e4fd7e26	https://tywqqefmllgseuvdvoia.supabase...	[null]	0	true	2025-07-15 22:59:07.741379
2	4923bc3e-207f-43ea-9fce-0aa61d385029	39f5d135-a1f3-434f-9ff8-ce34e4fd7e26	https://tywqqefmllgseuvdvoia.supabase...	[null]	1	false	2025-07-15 22:59:07.753696
3	d456b708-e0ab-42e8-bd82-cb216759f0...	39f5d135-a1f3-434f-9ff8-ce34e4fd7e26	https://tywqqefmllgseuvdvoia.supabase...	[null]	2	false	2025-07-15 22:59:07.76166
4	ce8ecbed-481b-401e-96ee-709130ea7c...	038a03ac-7383-4f96-bbe9-f5c7b2bc356e	https://tywqqefmllgseuvdvoia.supabase...	[null]	0	true	2025-07-16 00:05:50.282862
5	0d8484d0-c948-4ce4-8c4c-ff40c474b667	038a03ac-7383-4f96-bbe9-f5c7b2bc356e	https://tywqqefmllgseuvdvoia.supabase...	[null]	1	false	2025-07-16 00:05:50.291849
6	cb0287bf-558b-4d75-bca2-9e3f81a4ae38	038a03ac-7383-4f96-bbe9-f5c7b2bc356e	https://tywqqefmllgseuvdvoia.supabase...	[null]	2	false	2025-07-16 00:05:50.30091
7	a407ea25-d66f-44de-9c39-a90b19abe2e1	b0b981cf-1859-4439-b6c1-2ecf7557307c	https://tywqqefmllgseuvdvoia.supabase...	[null]	0	true	2025-07-16 00:13:33.894445
8	059570fd-1e8a-4517-bb01-7f10261b30d9	b0b981cf-1859-4439-b6c1-2ecf7557307c	https://tywqqefmllgseuvdvoia.supabase...	[null]	1	false	2025-07-16 00:13:33.906443
9	2c28999f-cae2-48d5-9b20-7d55a488f2be	b0b981cf-1859-4439-b6c1-2ecf7557307c	https://tywqqefmllgseuvdvoia.supabase...	[null]	2	false	2025-07-16 00:13:33.918448

3.2.6 bookings Table Snapshot

	id [PK] uuid	package_id uuid	user_id uuid	promo_code_id uuid	status character varying (20)	payment_status character varying	booking_date timestamp with	cancellation_da timestamp with	cancellation_ text	created_at timestamp witho	updated_at timestamp w	total_amount numeric (10,2)	paid_amoun numeric (10,	discoun numeric
8	b7f8a9c0-2d...	f3b2a1d1-9e0...	cc63af5e-feb6-...	[null]	confirmed	partial	2025-07-09...	[null]	[null]	2025-07-16 0...	2025-07-...	13500.00	6750.00	0.00
9	c1d2e3f4-5a...	f3b2a1d1-9e0...	b9dc1d92-d3aa...	9c32f1d0-e...	cancelled	refunded	2025-06-15...	2025-06-18 ...	Schedul...	2025-07-16 0...	2025-07-...	14500.00	0.00	0.00
10	d4e5f6a7-8b...	f3b2a1d1-9e0...	a95e7e2b-1e4a...	8b56a2f1-3...	confirmed	paid	2025-07-04...	[null]	[null]	2025-07-16 0...	2025-07-...	9800.00	9000.00	800.00
11	f37c43be-7d...	b0b981cf-185...	5a85f0a3-6cb0-...	[null]	confirmed	paid	2025-07-14...	[null]	[null]	2025-07-14 2...	2025-07-...	3600.00	3600.00	0.00
12	0b44c318-98...	39f5d135-a1f...	5a85f0a3-6cb0-...	[null]	pending	pending	2025-07-15...	[null]	[null]	2025-07-15 2...	2025-07-...	4000.00	0.00	0.00
13	08fb6eb3-dc...	038a03ac-738...	5a85f0a3-6cb0-...	[null]	pending	pending	2025-07-15...	[null]	[null]	2025-07-15 2...	2025-07-...	1500.00	0.00	0.00
14	3e9383e6-3d...	39f5d135-a1f...	c361a0ec-603e...	[null]	pending	pending	2025-07-16...	[null]	[null]	2025-07-16 0...	2025-07-...	4000.00	0.00	0.00
15	a1b877f3-6e...	038a03ac-738...	c361a0ec-603e...	[null]	pending	pending	2025-07-16...	[null]	[null]	2025-07-16 0...	2025-07-...	1500.00	0.00	0.00
16	2247d110-b1...	b0b981cf-185...	c361a0ec-603e...	[null]	pending	pending	2025-07-16...	[null]	[null]	2025-07-16 0...	2025-07-...	1500.00	0.00	0.00

3.2.7 destinations Table Snapshot

	id [PK] uuid	name character varying (255)	country character varying (100)	city character varying (100)	description text	best_time_to_visit character varying (1(	featured_image text	is_active boolean	created_at timestamp witho	updated_at timestamp without tim	created_by uuid
1	09e5b484-fd...	Cox's Bazar	Bangladesh	Chattogram	It's a goo...	December to Jan...	https://tywqqefmllgs...	true	2025-07-08 2...	2025-07-08 22:03:1...	123e456...
2	e9f85003-a9...	Sundarbans	Bangladesh	Khulna	The Sund...	October to March	https://tywqqefmllgs...	true	2025-07-15 2...	2025-07-15 22:49:2...	3a24275f...
3	7cce5e46-62...	Sylhet	Bangladesh	Sylhet	Sylhet, lo...	October to March	https://tywqqefmllgs...	true	2025-07-16 0...	2025-07-16 00:09:1...	3a24275f...
4	d1b5ea5a-12...	Cox's Bazar Beach	Bangladesh	Cox's Bazar	Longest ...	November - Febr...	https://example.com...	true	2025-07-16 0...	2025-07-16 01:33:2...	a1a1a1a1...
5	d1b5ea5a-12...	Mangrove Forest	Bangladesh	Khulna	Largest ...	October - March	https://example.com...	true	2025-07-16 0...	2025-07-16 01:33:2...	a1a1a1a1...
6	d1b5ea5a-12...	Sajek Valley	Bangladesh	Rangamati	Hill area ...	November - April	https://example.com...	true	2025-07-16 0...	2025-07-16 01:33:2...	a1a1a1a1...
7	d1b5ea5a-12...	Bandarban	Bangladesh	Bandarban	Trekking ...	November - Febr...	https://example.com...	true	2025-07-16 0...	2025-07-16 01:33:2...	a1a1a1a1...
8	d1b5ea5a-12...	Saint Martin's	Bangladesh	Teknaf	Small isl...	November - March	https://example.com...	true	2025-07-16 0...	2025-07-16 01:33:2...	a1a1a1a1...
9	d1b5ea5a-12...	Nilgiri	Bangladesh	Bandarban	Scenic m...	October - March	https://example.com...	true	2025-07-16 0...	2025-07-16 01:33:2...	a1a1a1a1...

3.2.8 promo-codes Table Snapshot

	id [PK] uuid	code character varying (	description text	discount_type character varying (2	discount_valu double precisi	minimum_amou double precisio	maximum_discount double precision	start_date date	expiry_date date	usage_limit integer	used_cou integer	is_active boolean	created_at timestamp without	updated_ timestan	created_ uuid
1	0115e54...	SAVE20	Wellcome	percentage	20	1500	2000	2025-07-08	2025-08-07	2	0	true	2025-07-08 18:...	2025...	123...
2	e101bc1...	SUMMER25	25% off summer ...	percentage	25	1000	500	2025-07-01	2025-08-31	100	12	true	2025-07-16 01:...	2025...	a1a...
3	e101bc1...	WELCOME100	৳100 off for new ...	fixed	100	0	[null]	2025-01-01	2025-12-31	500	80	true	2025-07-16 01:...	2025...	a1a...
4	e101bc1...	EARLYBIRD	Flat 10% for early...	percentage	10	2000	300	2025-06-01	2025-09-01	300	22	true	2025-07-16 01:...	2025...	a1a...
5	e101bc1...	MONSOON50	৳50 off on monso...	fixed	50	500	100	2025-07-10	2025-09-10	200	30	true	2025-07-16 01:...	2025...	a1a...
6	e101bc1...	REFER20	20% off via referr...	percentage	20	1000	300	2025-04-01	2025-12-31	1000	200	true	2025-07-16 01:...	2025...	a1a...
7	e101bc1...	BLACKFRIDAY	Black Friday deal:...	fixed	300	3000	[null]	2025-11-25	2025-11-30	100	50	true	2025-07-16 01:...	2025...	a1a...
8	e101bc1...	SPRING15	Spring trip 15% off	percentage	15	1500	400	2025-03-01	2025-05-31	250	99	true	2025-07-16 01:...	2025...	a1a...
9	e101bc1...	FAMILYFUN	৳200 off for grou...	fixed	200	3000	200	2025-05-01	2025-12-01	300	50	true	2025-07-16 01:...	2025...	a1a...

3.2.9 blogs Table Snapshot

	id [PK] uuid	title character varying	author_id uuid	content text	excerpt text	status character var	published_at timestamp without ti	category character varyi	cover_image text	is_featured boolean	created_at timestamp without time	updated_at timestamp
5	b101ec1a-1001-4bc...	A Weekend in ...	144be6a9-8...	Experience Sajek fr...	Tips for travelin...	published	2025-07-02 00:0...	Nature	https://example.c...	false	2025-07-16 01:40:5...	2025-07...
6	b101ec1a-1001-4bc...	Saint Martin's I...	c361a0ec-6...	How to reach, what ...	Ultimate travel g...	published	2025-06-28 00:0...	Island	https://example.c...	true	2025-07-16 01:40:5...	2025-07...
7	b101ec1a-1001-4bc...	Best Adventur...	1f37c497-2...	Nilgiri, Nafakhum a...	Hidden trekking ...	published	2025-07-03 00:0...	Adventure	https://example.c...	false	2025-07-16 01:40:5...	2025-07...
8	b101ec1a-1001-4bc...	Budget Travel ...	d72a453b-e...	How to save while tr...	Student discoun...	published	2025-06-30 00:0...	Tips	https://example.c...	false	2025-07-16 01:40:5...	2025-07...
9	b101ec1a-1001-4bc...	Monsoon Dest...	5e99a3c8-1...	Best rainy season s...	When to travel i...	published	2025-07-05 00:0...	Seasonal	https://example.c...	false	2025-07-16 01:40:5...	2025-07...
10	b101ec1a-1001-4bc...	Travel Essenti...	f441d68e-0...	What to pack before...	Don't forget the...	draft	[null]	Tips	https://example.c...	false	2025-07-16 01:40:5...	2025-07...
11	b101ec1a-1001-4bc...	Exploring Jafl...	cc63af5e-fe...	Tea gardens, rivers, ...	What to do in Ja...	published	2025-07-06 00:0...	Culture	https://example.c...	true	2025-07-16 01:40:5...	2025-07...
12	b101ec1a-1001-4bc...	7 Mistakes Fir...	b9dc1d92-d...	Avoid these commo...	Pack smart, pla...	published	2025-07-07 00:0...	Advice	https://example.c...	false	2025-07-16 01:40:5...	2025-07...
13	b101ec1a-1001-4bc...	How to Plan a ...	a95e7e2b-1...	Organizing group to...	Roles, budgetin...	published	2025-07-08 00:0...	Group Travel	https://example.c...	false	2025-07-16 01:40:5...	2025-07...

4 Query Examples

4.1 Basic Queries with Joins

4.1.1 Cross Product Example

```sql
-- Query: Find all possible pairs of users and packages available in
    the system.
SELECT u.uid, u.full_name, p.id AS package_id, p.title
FROM users u
CROSS JOIN packages p;
```

Listing 10: Cross Product - All User-Packages Combinations

Relational Algebra:

$$\pi_{\text{uid,full\_name,id,title}} (\text{users} \times \text{packages})$$

Query Output:

	uid uuid	full_name character varying (255)	package_id uuid	title character varying (255)
5	5a85f0a3-6cb0-4578-b6ec-70a3813609...	Mehedi Hasan	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111002	Sajek Cloudy Escape
6	5a85f0a3-6cb0-4578-b6ec-70a3813609...	Mehedi Hasan	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111003	Bandarban Trekking
7	5a85f0a3-6cb0-4578-b6ec-70a3813609...	Mehedi Hasan	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111004	Saint Martin Getaway
8	5a85f0a3-6cb0-4578-b6ec-70a3813609...	Mehedi Hasan	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111005	Jaflong Nature Tour
9	144be6a9-8794-46d6-9a9c-86ed44fb0b...	New	39f5d135-a1f3-434f-9ff8-ce34e4fd7e26	Discover Sundarbans Adventure
10	144be6a9-8794-46d6-9a9c-86ed44fb0b...	New	038a03ac-7383-4f96-bbe9-f5c7b2bc356e	Relaxing Beach Holiday at Cox's Bazar
11	144be6a9-8794-46d6-9a9c-86ed44fb0b...	New	b0b981cf-1859-4439-b6c1-2ecf7557307c	Mystic Sylhet Tea Garden Retreat
12	144be6a9-8794-46d6-9a9c-86ed44fb0b...	New	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111001	Cox's Beach Bonanza
13	144be6a9-8794-46d6-9a9c-86ed44fb0b...	New	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111002	Sajek Cloudy Escape
14	144be6a9-8794-46d6-9a9c-86ed44fb0b...	New	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111003	Bandarban Trekking
15	144be6a9-8794-46d6-9a9c-86ed44fb0b...	New	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111004	Saint Martin Getaway
16	144be6a9-8794-46d6-9a9c-86ed44fb0b...	New	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111005	Jaflong Nature Tour
17	c361a0ec-603e-482a-b52e-795d12cadd...	Abu Bakar Siddique	39f5d135-a1f3-434f-9ff8-ce34e4fd7e26	Discover Sundarbans Adventure
18	c361a0ec-603e-482a-b52e-795d12cadd...	Abu Bakar Siddique	038a03ac-7383-4f96-bbe9-f5c7b2bc356e	Relaxing Beach Holiday at Cox's Bazar

4.1.2 Left Outer Join Example

```sql
-- Query: Retrieve all packages along with their booking information (if any).
-- Show even those packages that have never been booked.
SELECT p.id AS package_id, p.title, b.id AS booking_id, b.user_id, b.status
FROM packages p
LEFT OUTER JOIN bookings b ON p.id = b.package_id;
```

Listing 11: Left Outer Join - All Packages With or Without Bookings

Relational Algebra:

$$\pi_{\text{p.id, p.title, b.id, b.user\_id, b.status}} (\text{packages} \bowtie_{\text{packages.id=bookings.package\_id}} \text{bookings})$$

Query Output:

	package_id uuid	title character varying (255)	booking_id uuid	user_id uuid	status character varying (20)
1	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Cox's Beach Bonanza	a9f2a1d1-3e9c-4f5b-b9c1-6d7e8f111001	5a85f0a3-6cb0-4578-b6ec-70a381360965	confirmed
2	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Sajek Cloudy Escape	b0c8b8d9-5e6f-4e8c-b0d3-1a2b3c4d5002	144be6a9-8794-46d6-9a9c-86ed44fb0b6f	pending
3	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Bandarban Trekking	c7a2f0d1-1b2c-4d5e-9f8a-0b1c2d3e4003	c361a0ec-603e-482a-b52e-795d12caddf0	cancelled
4	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Saint Martin Getaway	d5b7f2a9-7f5e-4a9b-8c7d-1e2f3a4b6004	1f37c497-2b0a-4c20-b5f3-88a3ad122001	confirmed
5	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Jaflong Nature Tour	e0c9a3d2-8d7b-4a8f-bcd1-2e3f4a5b7005	d72a453b-e1cc-4bd8-84ea-1b66f0202002	confirmed
6	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Cox's Beach Bonanza	f2d3b1a0-9c4e-4a8d-b9d2-3c4e5f6a8006	5e99a3c8-19e2-4a90-8049-344fcfb32003	confirmed
7	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Sajek Cloudy Escape	a5e6c7d8-1f2a-4b3c-9d4e-5f6a7b8c9007	f441d68e-0a31-4b20-93d1-7d1c2e440004	pending
8	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Saint Martin Getaway	b7f8a9c0-2d3e-4f5a-8b9c-0d1e2f3a4008	cc63af5e-feb6-4873-b219-507ac8110005	confirmed
9	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Bandarban Trekking	c1d2e3f4-5a6b-7c8d-9e0f-1a2b3c4d5009	b9dc1d92-d3aa-4db0-9880-91d7ed5500...	cancelled
10	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Jaflong Nature Tour	d4e5f6a7-8b9c-0d1e-2f3a-4b5c6d7e8010	a95e7e2b-1e4a-41e9-86e1-7a598ea600...	confirmed
11	b0b981cf-1859-4439-b6c1-2ecf7557307c	Mystic Sylhet Tea Garden Retreat	f37c43be-7d5e-4153-9b83-9be8d6133ea2	5a85f0a3-6cb0-4578-b6ec-70a381360965	confirmed
12	39f5d135-a1f3-434f-9ff8-ce34e4fd7e26	Discover Sundarbans Adventure	0b44c318-9822-4b8a-8651-f284b0403676	5a85f0a3-6cb0-4578-b6ec-70a381360965	pending
13	038a03ac-7383-4f96-bbe9-f5c7b2bc356e	Relaxing Beach Holiday at Cox's Bazar	08fb6eb3-dc56-4bdd-9305-078ced980f47	5a85f0a3-6cb0-4578-b6ec-70a381360965	pending
14	39f5d135-a1f3-434f-9ff8-ce34e4fd7e26	Discover Sundarbans Adventure	3e9383e6-3d1b-4449-96b5-72664e683d...	c361a0ec-603e-482a-b52e-795d12caddf0	pending

4.1.3 Join with ON Example

```sql
-- Query: Retrieve booking info with package title and price
SELECT b.id AS booking_id, b.user_id, p.title, p.price
FROM bookings b
JOIN packages p ON b.package_id = p.id;
```

Listing 12: Join with ON - Join Bookings and Packages by Package ID

Relational Algebra:

$$\pi_{\text{b.id, b.user\_id, p.title, p.price}} (\text{bookings} \bowtie_{\text{bookings.package\_id=packages.id}} \text{packages})$$

Query Output:

	booking_id uuid	user_id uuid	title character varying (255)	price numeric (10,2)
1	a9f2a1d1-3e9c-4f5b-b9c1-6d7e8f111001	5a85f0a3-6cb0-4578-b6ec-70a381360965	Cox's Beach Bonanza	12500.00
2	b0c8b8d9-5e6f-4e8c-b0d3-1a2b3c4d5002	144be6a9-8794-46d6-9a9c-86ed44fb0b6f	Sajek Cloudy Escape	10500.00
3	c7a2f0d1-1b2c-4d5e-9f8a-0b1c2d3e4003	c361a0ec-603e-482a-b52e-795d12caddf0	Bandarban Trekking	14500.00
4	d5b7f2a9-7f5e-4a9b-8c7d-1e2f3a4b6004	1f37c497-2b0a-4c20-b5f3-88a3ad122001	Saint Martin Getaway	13500.00
5	e0c9a3d2-8d7b-4a8f-bcd1-2e3f4a5b7005	d72a453b-e1cc-4bd8-84ea-1b66f0202002	Jaflong Nature Tour	9800.00
6	f2d3b1a0-9c4e-4a8d-b9d2-3c4e5f6a8006	5e99a3c8-19e2-4a90-8049-344fcfb32003	Cox's Beach Bonanza	12500.00
7	a5e6c7d8-1f2a-4b3c-9d4e-5f6a7b8c9007	f441d68e-0a31-4b20-93d1-7d1c2e440004	Sajek Cloudy Escape	10500.00
8	b7f8a9c0-2d3e-4f5a-8b9c-0d1e2f3a4008	cc63af5e-feb6-4873-b219-507ac8110005	Saint Martin Getaway	13500.00
9	c1d2e3f4-5a6b-7c8d-9e0f-1a2b3c4d5009	b9dc1d92-d3aa-4db0-9880-91d7ed5500…	Bandarban Trekking	14500.00
10	d4e5f6a7-8b9c-0d1e-2f3a-4b5c6d7e8010	a95e7e2b-1e4a-41e9-86e1-7a598ea600…	Jaflong Nature Tour	9800.00
11	f37c43be-7d5e-4153-9b83-9be8d6133ea2	5a85f0a3-6cb0-4578-b6ec-70a381360965	Mystic Sylhet Tea Garden Retreat	1500.00
12	0b44c318-9822-4b8a-8651-f284b0403676	5a85f0a3-6cb0-4578-b6ec-70a381360965	Discover Sundarbans Adventure	4000.00
13	08fb6eb3-dc56-4bdd-9305-078ced980f47	5a85f0a3-6cb0-4578-b6ec-70a381360965	Relaxing Beach Holiday at Cox's Bazar	1500.00
14	3e9383e6-3d1b-4449-96b5-72664e683d…	c361a0ec-603e-482a-b52e-795d12caddf0	Discover Sundarbans Adventure	4000.00

4.2 Nested Subqueries

4.2.1 Subquery with ANY Example

```sql
-- Query: Find packages where the price is less than any paid amount in
    bookings
SELECT p.id, p.title, p.price
FROM packages p
WHERE p.price < ANY (
    SELECT b.paid_amount
    FROM bookings b
);
```

Listing 13: Find packages priced less than any booking's paid amount

Relational Algebra:

Expressing ANY in relational algebra is not straightforward, but the query conceptually means:

$$\{p \in \text{packages} \mid \exists b \in \text{bookings such that } p.price < b.paid\_amount\}$$

Or as a selection:

$$\sigma_{\text{price<paid\_amount}}(\text{packages} \times \text{bookings})$$

followed by a projection on package attributes with the condition that **at least one booking's paid amount is greater than package price**.

Query Output:

	id [PK] uuid	title character varying (255)	price numeric (10,2)
1	39f5d135-a1f3-434f-9ff8-ce34e4fd7e26	Discover Sundarbans Adventure	4000.00
2	038a03ac-7383-4f96-bbe9-f5c7b2bc356e	Relaxing Beach Holiday at Cox's Bazar	1500.00
3	b0b981cf-1859-4439-b6c1-2ecf7557307c	Mystic Sylhet Tea Garden Retreat	1500.00
4	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111002	Sajek Cloudy Escape	10500.00
5	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111005	Jaflong Nature Tour	9800.00

4.2.2 Subquery with SOME Example

```sql
-- Query: Find packages priced less than some packages priced over 1000
SELECT p1.id, p1.title, p1.price
FROM packages p1
WHERE p1.price < SOME (
    SELECT p2.price
    FROM packages p2
    WHERE p2.price > 1000
);
```

Listing 14: Find packages cheaper than some packages with high price

Relational Algebra:

$$\pi_{\text{id, title, price}}\Big(\sigma_{p1.price<p2.price \wedge p2.price>1000}(packages_{p1} \times packages_{p2})\Big)$$

Query Output:

	id [PK] uuid	title character varying (255)	price numeric (10,2)
1	39f5d135-a1f3-434f-9ff8-ce34e4fd7e26	Discover Sundarbans Adventure	4000.00
2	038a03ac-7383-4f96-bbe9-f5c7b2bc356e	Relaxing Beach Holiday at Cox's Bazar	1500.00
3	b0b981cf-1859-4439-b6c1-2ecf7557307c	Mystic Sylhet Tea Garden Retreat	1500.00
4	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111001	Cox's Beach Bonanza	12500.00
5	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111002	Sajek Cloudy Escape	10500.00
6	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111004	Saint Martin Getaway	13500.00
7	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111005	Jaflong Nature Tour	9800.00

4.2.3 Subquery with ALL Example

```sql
-- Query: Find packages priced less than or equal to all packages
   priced over 500
SELECT p1.id, p1.title, p1.price
FROM packages p1
WHERE p1.price <= ALL (
    SELECT p2.price
    FROM packages p2
    WHERE p2.price > 500
);
```

Listing 15: Find packages priced less than or equal to all packages priced over 500

Relational Algebra:

$$\pi_{\text{id, title, price}}\left(\sigma_{\forall p2(p2.price>500 \implies p1.price\leq p2.price)}(packages_{p1})\right)$$

Query Output:

	id [PK] uuid	title character varying (255)	price numeric (10,2)
1	038a03ac-7383-4f96-bbe9-f5c7b2bc356e	Relaxing Beach Holiday at Cox's Bazar	1500.00
2	b0b981cf-1859-4439-b6c1-2ecf7557307c	Mystic Sylhet Tea Garden Retreat	1500.00

4.2.4 Subquery with EXISTS Example

```sql
-- Query: Find packages for which there exists at least one booking
SELECT p.id, p.title, p.price
FROM packages p
WHERE EXISTS (
    SELECT 1
    FROM bookings b
    WHERE b.package_id = p.id
);
```

Listing 16: Find packages that have at least one booking

Relational Algebra:

$$\pi_{\text{id, title, price}}\left(\sigma_{\exists b\in bookings, b.package\_id=p.id}(packages)\right)$$

Query Output:

	id [PK] uuid	title character varying (255)	price numeric (10,2)
1	39f5d135-a1f3-434f-9ff8-ce34e4fd7e26	Discover Sundarbans Adventure	4000.00
2	038a03ac-7383-4f96-bbe9-f5c7b2bc356e	Relaxing Beach Holiday at Cox's Bazar	1500.00
3	b0b981cf-1859-4439-b6c1-2ecf7557307c	Mystic Sylhet Tea Garden Retreat	1500.00
4	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111001	Cox's Beach Bonanza	12500.00
5	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111002	Sajek Cloudy Escape	10500.00
6	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111003	Bandarban Trekking	14500.00
7	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111004	Saint Martin Getaway	13500.00
8	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111005	Jaflong Nature Tour	9800.00

4.3 Subqueries in Different Clauses

4.3.1 Subquery in FROM Clause Example

```sql
-- Query: Find the average paid amount per package using a subquery in
   FROM clause
SELECT sub.package_id, sub.avg_paid_amount
FROM (
    SELECT b.package_id, AVG(b.paid_amount) AS avg_paid_amount
    FROM bookings b
```

```
    GROUP BY b.package_id
) sub
JOIN packages p ON sub.package_id = p.id;
```

Listing 17: Calculate average booking amount per package

Relational Algebra:

1. Compute the aggregation (average) grouped by package-id:

$$\gamma_{\text{package\_id};\, \text{AVG(paid\_amount)} \to \text{avg\_paid\_amount}}(\text{bookings})$$

2. Join the result with packages on package-id = id:

$$\pi_{\text{sub.package\_id},\, \text{sub.avg\_paid\_amount}}\left(\gamma_{\text{package\_id};\, \text{AVG(paid\_amount)}}(\text{bookings}) \bowtie_{\text{sub.package\_id}=\text{packages.id}} \text{packages}\right)$$

Query Output:

	package_id uuid	avg_paid_amount numeric
1	39f5d135-a1f3-434f-9ff8-ce34e4fd7e26	0.00000000000000000000
2	038a03ac-7383-4f96-bbe9-f5c7b2bc356e	0.00000000000000000000
3	b0b981cf-1859-4439-b6c1-2ecf7557307c	1800.0000000000000000
4	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111001	6000.0000000000000000
5	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111002	0.00000000000000000000
6	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111003	0.00000000000000000000
7	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111004	6875.0000000000000000
8	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111005	9400.0000000000000000

4.3.2 Subquery in WHERE Clause Example

```
-- Query: Find packages that have at least one booking with paid_amount
    > 1000
SELECT p.id, p.title, p.price
FROM packages p
WHERE p.id IN (
    SELECT b.package_id
    FROM bookings b
    WHERE b.paid_amount > 1000
);
```

Listing 18: Find packages that have bookings with paid amount greater than 1000

Relational Algebra:

1. Select bookings with paid amount ¿ 1000:

$$\sigma_{\text{paid\_amount}>1000}(\text{bookings})$$

2. Project package IDs from these bookings:

$$\pi_{\text{package\_id}}\left(\sigma_{\text{paid\_amount}>1000}(\text{bookings})\right)$$

3. Select packages whose id is in the set of package IDs:

$$\sigma_{\text{id}\in\pi_{\text{package\_id}}(\sigma_{\text{paid\_amount}>1000}(\text{bookings}))}(\text{packages})$$

Query Output:

	id [PK] uuid	title character varying (255)	price numeric (10,2)
1	b0b981cf-1859-4439-b6c1-2ecf7557307c	Mystic Sylhet Tea Garden Retreat	1500.00
2	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111001	Cox's Beach Bonanza	12500.00
3	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111004	Saint Martin Getaway	13500.00
4	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111005	Jaflong Nature Tour	9800.00

4.3.3 Subquery in SELECT Clause Example

```sql
-- Query: List packages with the number of bookings for each
SELECT p.id, p.title,
    (SELECT COUNT(*)
     FROM bookings b
     WHERE b.package_id = p.id) AS booking_count
FROM packages p;
```

Listing 19: Show packages with the count of bookings

Relational Algebra:

1. Count bookings per package:

$$\gamma_{\text{package\_id};\,COUNT(*)\to\text{booking\_count}}(\text{bookings})$$

2. Left join packages with the aggregated counts (to include packages with zero bookings):

$$packages \;\; ⟕_{packages.id=bookings.package\_id} \;\; \gamma_{\text{package\_id};\,COUNT(*)}(\text{bookings})$$

3. Project package id, title, and booking count:

$$\pi_{id,\,title,\,booking\_count}(\ldots)$$

Query Output:

	id [PK] uuid	title character varying (255)	booking_count bigint
1	39f5d135-a1f3-434f-9ff8-ce34e4fd7e26	Discover Sundarbans Adventure	2
2	038a03ac-7383-4f96-bbe9-f5c7b2bc356e	Relaxing Beach Holiday at Cox's Bazar	2
3	b0b981cf-1859-4439-b6c1-2ecf7557307c	Mystic Sylhet Tea Garden Retreat	2
4	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111001	Cox's Beach Bonanza	2
5	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111002	Sajek Cloudy Escape	2
6	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111003	Bandarban Trekking	2
7	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111004	Saint Martin Getaway	2
8	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111005	Jaflong Nature Tour	2

4.4 Advanced Clauses

4.4.1 ORDER BY Clause Example

```sql
-- Query: Show all active packages sorted by highest price first
SELECT id, title, price
FROM packages
WHERE is_active = true
ORDER BY price DESC;
```

Listing 20: List packages ordered by price descending

Relational Algebra:

1. Select only active packages:

$$\sigma_{\text{is\_active}=\text{true}}(\text{packages})$$

2. Project relevant attributes:

$$\pi_{\text{id, title, price}}(\ldots)$$

3. Apply sorting by price in descending order (denoted as τ):

$$\tau_{\text{price desc}}(\pi_{\text{id, title, price}}(\sigma_{\text{is\_active}=\text{true}}(\text{packages})))$$

Query Output:

	id [PK] uuid	title character varying (255)	price numeric (10,2)
1	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Bandarban Trekking	14500.00
2	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Saint Martin Getaway	13500.00
3	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Cox's Beach Bonanza	12500.00
4	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Sajek Cloudy Escape	10500.00
5	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Jaflong Nature Tour	9800.00
6	39f5d135-a1f3-434f-9ff8-ce34e4fd7e26	Discover Sundarbans Adventure	4000.00
7	b0b981cf-1859-4439-b6c1-2ecf7557307c	Mystic Sylhet Tea Garden Retreat	1500.00
8	038a03ac-7383-4f96-bbe9-f5c7b2bc356e	Relaxing Beach Holiday at Cox's Bazar	1500.00

4.4.2 GROUP BY Clause Example

```sql
-- Query: Find total revenue (sum of paid_amount) grouped by package
SELECT package_id, SUM(paid_amount) AS total_revenue
FROM bookings
GROUP BY package_id;
```

Listing 21: Total revenue generated per package

Relational Algebra:

1. Group by package-id and aggregate SUM(paid-amount):

$$\gamma_{\text{package\_id; SUM(paid\_amount)}\rightarrow\text{total\_revenue}}(\text{bookings})$$

2. Project the result (optional if already just these attributes):

$$\pi_{\text{package\_id, total\_revenue}}(\dots)$$

Query Output:

	package_id uuid	total_revenue numeric
1	038a03ac-7383-4f96-bbe9-f5c7b2bc356e	0.00
2	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	18800.00
3	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	13750.00
4	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	12000.00
5	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	0.00
6	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	0.00
7	39f5d135-a1f3-434f-9ff8-ce34e4fd7e26	0.00
8	b0b981cf-1859-4439-b6c1-2ecf7557307c	3600.00

4.4.3 HAVING Clause Example

```sql
-- Query: Show packages that generated more than 10,000 in total
   revenue
SELECT package_id, SUM(paid_amount) AS total_revenue
FROM bookings
GROUP BY package_id
HAVING SUM(paid_amount) > 10000;
```

Listing 22: Packages with total revenue over 10

Relational Algebra:

1. Group by package-id and compute SUM(paid-amount):

$$G := \gamma_{\text{package\_id; SUM(paid\_amount)}\to\text{total\_revenue}}(\text{bookings})$$

2. Apply selection on the grouped result:

$$\sigma_{\text{total\_revenue}>10000}(G)$$

3. Project final result:

$$\pi_{\text{package\_id, total\_revenue}}(\dots)$$

Query Output:

	package_id uuid	total_revenue numeric
1	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111005	18800.00
2	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111004	13750.00
3	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a111001	12000.00

4.4.4 WITH Clause (CTE) Example

```
-- Query: Use a CTE to calculate total revenue per package and join
   with packages
WITH revenue_per_package AS (
    SELECT package_id, SUM(paid_amount) AS total_revenue
    FROM bookings
    GROUP BY package_id
)
SELECT p.id, p.title, COALESCE(r.total_revenue, 0) AS total_revenue
FROM packages p
LEFT JOIN revenue_per_package r ON p.id = r.package_id;
```

Listing 23: List packages with their total revenue using CTE

Relational Algebra:

1. Compute revenue per package as a temporary relation R:

$$R := \gamma_{\text{package\_id};\, \text{SUM(paid\_amount)} \to \text{total\_revenue}}(\text{bookings})$$

2. Left outer join with packages to include all packages:

$$J := \text{packages} \;\; ⟕_{\text{packages.id}=R.\text{package\_id}} \;\; R$$

3. Project the desired attributes:

$$\pi_{\text{id, title, total\_revenue}}(J)$$

Query Output:

	id [PK] uuid	title character varying (255)	total_revenue numeric
1	038a03ac-7383-4f96-bbe9-f5c7b2bc356e	Relaxing Beach Holiday at Cox's Bazar	0.00
2	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Jaflong Nature Tour	18800.00
3	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Saint Martin Getaway	13750.00
4	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Cox's Beach Bonanza	12000.00
5	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Sajek Cloudy Escape	0.00
6	f3b2a1d1-9e0c-4b5d-9c51-1c2e3a1110...	Bandarban Trekking	0.00
7	39f5d135-a1f3-434f-9ff8-ce34e4fd7e26	Discover Sundarbans Adventure	0.00
8	b0b981cf-1859-4439-b6c1-2ecf7557307c	Mystic Sylhet Tea Garden Retreat	3600.00

4.5 String and Set Operations

4.5.1 String Operations Example 4

```sql
-- Query: Select users whose full name contains 'Bakar' as a whole word
SELECT full_name
FROM users
WHERE full_name ILIKE '% Bakar %';
```

Listing 24: Find users with the word 'Bakar' in their name

Relational Algebra (Conceptual):

1. Use selection with a string match condition:

$$\sigma_{\text{full\_name ILIKE '\% Bakar \%'}}(\text{users})$$

2. Project only the full name:

$$\pi_{\text{full\_name}}(\ldots)$$

Query Output:

	full_name character varying (255)
1	Abu Bakar Siddique

4.6 Update and Delete Operations

4.6.1 Simple UPDATE Example

```sql
-- Query: Update full name of a specific user
UPDATE users
SET full_name = 'Farhan Ahmed'
WHERE email = 'alice@example.com';
```

Listing 25: Update a user's full name

Before Update:

	uid [PK] uuid	email character varying (255)	full_name character varying (255)	phone character varying (20)	passport character varying (255)
1	5a85f0a3-6cb0-4578-b6ec-70a3813609...	hasanmehedi26696@gmail.com	Mehedi Hasan	+8801319926696	12345678
2	144be6a9-8794-46d6-9a9c-86ed44fb0b...	jfwejrw@gmail.com	New	+8801319926696	2375983
3	c361a0ec-603e-482a-b52e-795d12cadd...	bojackabs@gmail.com	Abu Bakar Siddique	+880131992657	5789273982
4	1f37c497-2b0a-4c20-b5f3-88a3ad1220...	alice@example.com	Alice Wonderland	01710000001	P12345601
5	d72a453b-e1cc-4bd8-84ea-1b66f02020...	bob@example.com	Bob Builder	01710000002	P12345602
6	5e99a3c8-19e2-4a90-8049-344fcfb32003	charlie@example.com	Charlie Chaplin	01710000003	P12345603
7	f441d68e-0a31-4b20-93d1-7d1c2e4400...	diana@example.com	Diana Prince	01710000004	P12345604
8	cc63af5e-feb6-4873-b219-507ac8110005	ethan@example.com	Ethan Hunt	01710000005	P12345605
9	b9dc1d92-d3aa-4db0-9880-91d7ed550...	fiona@example.com	Fiona Gallagher	01710000006	P12345606

After Update:

	uid [PK] uuid	email character varying (255)	full_name character varying (255)	phone character varying (20)	passport character varying (255)
5	5e99a3c8-19e2-4a90-8049-344fcfb32003	charlie@example.com	Charlie Chaplin	01710000003	P12345603
6	f441d68e-0a31-4b20-93d1-7d1c2e4400...	diana@example.com	Diana Prince	01710000004	P12345604
7	cc63af5e-feb6-4873-b219-507ac8110005	ethan@example.com	Ethan Hunt	01710000005	P12345605
8	b9dc1d92-d3aa-4db0-9880-91d7ed5500...	fiona@example.com	Fiona Gallagher	01710000006	P12345606
9	a95e7e2b-1e4a-41e9-86e1-7a598ea600...	george@example.com	George Orwell	01710000007	P12345607
10	77f7cbbc-464e-46de-8826-0b4c10990008	hannah@example.com	Hannah Montana	01710000008	P12345608
11	89b01e4b-4690-4a4a-ae6e-87f117dd0009	ivan@example.com	Ivan Drago	01710000009	P12345609
12	3d65de21-cfc6-4f78-b83f-57eb2b2b0010	julia@example.com	Julia Roberts	01710000010	P12345610
13	1f37c497-2b0a-4c20-b5f3-88a3ad122001	alice@example.com	Farhan Ahmed	01710000001	P12345601

4.6.2 DELETE Query Example

```sql
-- Query: Delete a user from the system by email
DELETE FROM users
WHERE email = 'alice@example.com';
```

Listing 26: Delete a user by email

Before Deletion:

	uid [PK] uuid	email character varying (255)	full_name character varying (255)	phone character varying (20)	passport character varying (255)
5	5e99a3c8-19e2-4a90-8049-344fcfb32003	charlie@example.com	Charlie Chaplin	01710000003	P12345603
6	f441d68e-0a31-4b20-93d1-7d1c2e4400...	diana@example.com	Diana Prince	01710000004	P12345604
7	cc63af5e-feb6-4873-b219-507ac8110005	ethan@example.com	Ethan Hunt	01710000005	P12345605
8	b9dc1d92-d3aa-4db0-9880-91d7ed5500...	fiona@example.com	Fiona Gallagher	01710000006	P12345606
9	a95e7e2b-1e4a-41e9-86e1-7a598ea600...	george@example.com	George Orwell	01710000007	P12345607
10	77f7cbbc-464e-46de-8826-0b4c10990008	hannah@example.com	Hannah Montana	01710000008	P12345608
11	89b01e4b-4690-4a4a-ae6e-87f117dd0009	ivan@example.com	Ivan Drago	01710000009	P12345609
12	3d65de21-cfc6-4f78-b83f-57eb2b2b0010	julia@example.com	Julia Roberts	01710000010	P12345610
13	1f37c497-2b0a-4c20-b5f3-88a3ad122001	alice@example.com	Farhan Ahmed	01710000001	P12345601

After Deletion:

	uid [PK] uuid	email character varying (255)	full_name character varying (255)	phone character varying (20)	passport character varying (255)
4	d72a453b-e1cc-4bd8-84ea-1b66f02020...	bob@example.com	Bob Builder	01710000002	P12345602
5	5e99a3c8-19e2-4a90-8049-344fcfb32003	charlie@example.com	Charlie Chaplin	01710000003	P12345603
6	f441d68e-0a31-4b20-93d1-7d1c2e4400...	diana@example.com	Diana Prince	01710000004	P12345604
7	cc63af5e-feb6-4873-b219-507ac8110005	ethan@example.com	Ethan Hunt	01710000005	P12345605
8	b9dc1d92-d3aa-4db0-9880-91d7ed550...	fiona@example.com	Fiona Gallagher	01710000006	P12345606
9	a95e7e2b-1e4a-41e9-86e1-7a598ea600...	george@example.com	George Orwell	01710000007	P12345607
10	77f7cbbc-464e-46de-8826-0b4c109900...	hannah@example.com	Hannah Montana	01710000008	P12345608
11	89b01e4b-4690-4a4a-ae6e-87f117dd00...	ivan@example.com	Ivan Drago	01710000009	P12345609
12	3d65de21-cfc6-4f78-b83f-57eb2b2b0010	julia@example.com	Julia Roberts	01710000010	P12345610

4.7 Built-in Functions and Aggregates

4.7.1 Aggregate Functions Example

```sql
-- Query: Find total number of bookings and average paid amount per
    user
SELECT user_id,
       COUNT(*) AS total_bookings,
       AVG(paid_amount) AS avg_paid_amount
FROM bookings
GROUP BY user_id;
```

Listing 27: Count and average bookings per user

Relational Algebra:

1. Group bookings by user-id and compute aggregates:

$$\gamma_{\text{user\_id};\, COUNT(*)\rightarrow \text{total\_bookings},\, AVG(paid\_amount)\rightarrow \text{avg\_paid\_amount}}(\text{bookings})$$

2. Project relevant attributes (if needed):

$$\pi_{\text{user\_id},\, \text{total\_bookings},\, \text{avg\_paid\_amount}}(\ldots)$$

Query Output:

	user_id uuid	total_bookings bigint	avg_paid_amount numeric
2	b9dc1d92-d3aa-4db0-9880-91d7ed5500...	1	0.00000000000000000000
3	cc63af5e-feb6-4873-b219-507ac8110005	1	6750.00000000000000000
4	5a85f0a3-6cb0-4578-b6ec-70a381360965	4	3900.00000000000000000
5	a95e7e2b-1e4a-41e9-86e1-7a598ea60007	1	9000.00000000000000000
6	1f37c497-2b0a-4c20-b5f3-88a3ad122001	1	7000.00000000000000000
7	c361a0ec-603e-482a-b52e-795d12caddf0	4	0.00000000000000000000
8	d72a453b-e1cc-4bd8-84ea-1b66f0202002	1	9800.00000000000000000
9	5e99a3c8-19e2-4a90-8049-344fcfb32003	1	0.00000000000000000000
10	f441d68e-0a31-4b20-93d1-7d1c2e440004	1	0.00000000000000000000

4.7.2 Date and Time Functions Example

```sql
-- Query: Count bookings grouped by booking month for the year 2025
SELECT
    DATE_TRUNC('month', booking_date) AS booking_month,
    COUNT(*) AS bookings_count
FROM bookings
WHERE booking_date >= '2025-01-01' AND booking_date < '2026-01-01'
GROUP BY booking_month
ORDER BY booking_month;
```

Listing 28: Count bookings per month in 2025

Relational Algebra:

1. Select bookings in 2025:

$$\sigma_{\text{booking\_date} \geq \text{'2025-01-01'} \wedge \text{booking\_date} < \text{'2026-01-01'}}(\text{bookings})$$

2. Project booking month (via truncation) and count bookings per month (grouping):

$$\gamma_{\text{booking\_month}; COUNT(*) \rightarrow \text{bookings\_count}}(\dots)$$

3. Order results by month (sorting operator τ):

$$\tau_{\text{booking\_month}}(\dots)$$

Query Output:

	booking_month timestamp without time zone	bookings_count bigint
1	2025-06-01 00:00:00	2
2	2025-07-01 00:00:00	14

5 Views Implementation

5.1 View Creation

5.1.1 CREATE VIEW with Joins and Aggregates Example

```sql
CREATE VIEW package_summary AS
SELECT
    p.id,
    p.title,
    p.price,
    p.is_featured,
    d.name AS destination_name,
    d.country,
    d.city,
    COUNT(b.id) AS total_bookings,
    COALESCE(SUM(b.total_amount), 0) AS total_revenue,
    COALESCE(AVG(b.total_amount), 0) AS avg_booking_value,
    p.created_at
FROM packages p
JOIN destinations d ON p.destination_id = d.id
LEFT JOIN bookings b ON p.id = b.package_id AND b.status = 'confirmed'
WHERE p.is_active = true
GROUP BY
    p.id, p.title, p.price, p.is_featured,
    d.name, d.country, d.city,
    p.created_at;
```

Listing 29: Create a view summarizing package booking data with destination info

Relational Algebra:

1. Join packages and destinations:

$$J_1 := \text{packages} \bowtie_{\text{packages.destination\_id=destinations.id}} \text{destinations}$$

2. Left join with bookings filtered by confirmed status:

$$J_2 := J_1 \ \text{⟕}_{\text{packages.id=bookings.package\_id} \wedge \text{bookings.status='confirmed'}} \ \text{bookings}$$

3. Select only active packages:

$$\sigma_{\text{is\_active=true}}(J_2)$$

4. Group by relevant attributes and aggregate booking counts and sums:

$$\gamma_{\substack{p.id, p.title, p.price, p.is\_featured, \\ d.name, d.country, d.city, p.created\_at;}} \begin{cases} \text{COUNT}(b.id) \rightarrow \text{total\_bookings}, \\ \text{SUM}(b.total\_amount) \rightarrow \text{total\_revenue}, \\ \text{AVG}(b.total\_amount) \rightarrow \text{avg\_booking\_value} \end{cases} \quad (\ldots)$$

Query Output:

	id uuid	title character varying (255)	price numeric (10,2)	is_featured boolean	destination_name character varying (255)	country character varying (100)	city character varying (1	total_boo bigint	total_revenue numeric	avg_booking_ numeric	created_at timestamp without time zone
1	038a03ac-7383-...	Relaxing Beach Holiday ...	1500.00	false	Cox's Bazar	Bangladesh	Chattogram	0	0	0	2025-07-08 23:20:34.292412
2	39f5d135-a1f3-4...	Discover Sundarbans A...	4000.00	false	Sundarbans	Bangladesh	Khulna	0	0	0	2025-07-15 22:59:07.700933
3	b0b981cf-1859-4...	Mystic Sylhet Tea Garde...	1500.00	false	Sylhet	Bangladesh	Sylhet	1	3600.00	3600.000000	2025-07-08 22:08:47.930273
4	f3b2a1d1-9e0c-4...	Cox's Beach Bonanza	12500.00	true	Cox's Bazar Beach	Bangladesh	Cox's Bazar	2	25000.00	12500.00000	2025-07-16 01:37:25.036793
5	f3b2a1d1-9e0c-4...	Sajek Cloudy Escape	10500.00	true	Sajek Valley	Bangladesh	Rangamati	0	0	0	2025-07-16 01:37:25.036793
6	f3b2a1d1-9e0c-4...	Bandarban Trekking	14500.00	false	Bandarban	Bangladesh	Bandarban	0	0	0	2025-07-16 01:37:25.036793
7	f3b2a1d1-9e0c-4...	Saint Martin Getaway	13500.00	true	Saint Martin's	Bangladesh	Teknaf	2	27000.00	13500.00000	2025-07-16 01:37:25.036793
8	f3b2a1d1-9e0c-4...	Jaflong Nature Tour	9800.00	true	Jaflong	Bangladesh	Sylhet	2	19600.00	9800.000000	2025-07-16 01:37:25.036793

5.1.2 CREATE VIEW for User Activity Summary

```sql
CREATE VIEW user_activity_summary AS
SELECT
    u.uid,
    u.full_name,
    u.email,
    u.is_active,
    COUNT(b.id) AS total_bookings,
    COALESCE(SUM(b.total_amount), 0) AS total_spent,
    COALESCE(AVG(b.total_amount), 0) AS avg_spending,
    MAX(b.booking_date) AS last_booking_date,
    u.created_at AS registration_date
FROM users u
LEFT JOIN bookings b ON u.uid = b.user_id AND b.status = 'confirmed'
GROUP BY
    u.uid, u.full_name, u.email, u.is_active, u.created_at;
```

Listing 30: Create a view summarizing user booking activity

Relational Algebra:

1. Left join users with confirmed bookings:

$$J := \text{users} \ \bowtie_{\text{users.uid=bookings.user\_id} \wedge \text{bookings.status='confirmed'}} \ \text{bookings}$$

2. Group by user attributes, aggregate booking counts, sums, averages, and max date:

$$\gamma_{u.uid, u.full\_name, u.email, u.is\_active, u.created\_at;} \begin{cases} \text{COUNT}(b.id) \rightarrow \text{total\_bookings}, \\ \text{SUM}(b.total\_amount) \rightarrow \text{total\_spent}, \\ \text{AVG}(b.total\_amount) \rightarrow \text{avg\_spending}, \\ \text{MAX}(b.booking\_date) \rightarrow \text{last\_booking\_date} \end{cases} (J)$$

Query Output:

	uid uuid	full_name character varying (255)	email character varying (255)	is_active boolean	total_bookings bigint	total_spent numeric	avg_spending numeric	last_booking_date timestamp without ti	registration_date timestamp without time zone
2	1f37c497-2b0a-4c20-b5f3-88a3a...	Alice Wonderland	alice@example.com	true	1	13500.00	13500.000000(	2025-07-10 12:00...	2025-07-16 01:30:51.639812
3	3d65de21-cfc6-4f78-b83f-57eb2b...	Julia Roberts	julia@example.com	true	0	0	0	[null]	2025-07-16 01:30:51.639812
4	5a85f0a3-6cb0-4578-b6ec-70a38...	Mehedi Hasan	hasanmehedi26696@gmail.com	true	2	16100.00	8050.000000(	2025-07-14 20:16...	2025-07-08 21:47:38.12295
5	5e99a3c8-19e2-4a90-8049-344fcf...	Charlie Chaplin	charlie@example.com	true	1	12500.00	12500.000000(	2025-07-08 09:00...	2025-07-16 01:30:51.639812
6	77f7cbbc-464e-46de-8826-0b4c1...	Hannah Montana	hannah@example.com	true	0	0	0	[null]	2025-07-16 01:30:51.639812
7	89b01e4b-4690-4a4a-ae6e-87f11...	Ivan Drago	ivan@example.com	true	0	0	0	[null]	2025-07-16 01:30:51.639812
8	a95e7e2b-1e4a-41e9-86e1-7a598...	George Orwell	george@example.com	true	1	9800.00	9800.000000(	2025-07-04 10:30...	2025-07-16 01:30:51.639812
9	b9dc1d92-d3aa-4db0-9880-91d7e...	Fiona Gallagher	fiona@example.com	true	0	0	0	[null]	2025-07-16 01:30:51.639812
10	c361a0ec-603e-482a-b52e-795d1...	Abu Bakar Siddique	bojackabs@gmail.com	true	0	0	0	[null]	2025-07-16 01:03:50.238281

5.2 Queries Using Views

5.2.1 Using `package_summary` View

```sql
-- Query: Find top performing packages using view
SELECT
    title,
    destination_name,
    country,
    total_bookings,
    total_revenue
FROM package_summary
WHERE total_bookings > 0
ORDER BY total_revenue DESC, total_bookings DESC
LIMIT 10;
```

Listing 31: Find top performing packages using view

Relational Algebra (Conceptual from View):

Let P be the result of the package-summary view.

1. Select only packages with bookings:

$$\sigma_{\text{total\_bookings}>0}(P)$$

2. Project relevant columns:

$$\pi_{\text{title, destination\_name, country, total\_bookings, total\_revenue}}(\ldots)$$

3. Sort by total revenue and bookings (descending):

$$\tau_{\text{total\_revenue DESC, total\_bookings DESC}}(\ldots)$$

4. Limit to top 10 results:

$$\delta_{10}(\ldots)$$

Query Output:

	title character varying (255)	destination_name character varying (255)	country character varying (100)	total_bookings bigint	total_revenue numeric
1	Saint Martin Getaway	Saint Martin's	Bangladesh	2	27000.00
2	Cox's Beach Bonanza	Cox's Bazar Beach	Bangladesh	2	25000.00
3	Jaflong Nature Tour	Jaflong	Bangladesh	2	19600.00
4	Mystic Sylhet Tea Garden Retreat	Sylhet	Bangladesh	1	3600.00

5.2.2 Using `user_activity_summary` View

```sql
-- Query: Find high-value customers using view
SELECT
    full_name,
    email,
    total_bookings,
    total_spent,
    avg_spending
FROM user_activity_summary
WHERE total_spent > 2000
ORDER BY total_spent DESC;
```

Listing 32: Find high-value customers using view

Relational Algebra (Conceptual from View):

Let U be the result of the user-activity-summary view.

1. Select users who spent more than 2000:

$$\sigma_{\text{total\_spent}>2000}(U)$$

2. Project relevant user activity attributes:

$$\pi_{\text{full\_name, email, total\_bookings, total\_spent, avg\_spending}}(\dots)$$

3. Sort by total spending in descending order:

$$\tau_{\text{total\_spent DESC}}(\dots)$$

Query Output:

	full_name character varying (255)	email character varying (255)	total_bookings bigint	total_spent numeric	avg_spending numeric
1	Mehedi Hasan	hasanmehedi26696@gmail.com	2	16100.00	8050.0000000000000000
2	Alice Wonderland	alice@example.com	1	13500.00	13500.0000000000000000
3	Ethan Hunt	ethan@example.com	1	13500.00	13500.0000000000000000
4	Charlie Chaplin	charlie@example.com	1	12500.00	12500.0000000000000000
5	George Orwell	george@example.com	1	9800.00	9800.0000000000000000
6	Bob Builder	bob@example.com	1	9800.00	9800.0000000000000000

6 Functional Dependencies and Normalization

6.1 Functional Dependencies Analysis

This section analyzes the functional dependencies (FDs) for all nine tables in the VoyageVista database system, examining how attributes depend on primary keys and other determinants.

6.1.1 Users Table Functional Dependencies

Table Schema: `users(uid, email, full_name, phone, passport, password_hash, is_active, bookings_count, created_at, updated_at)`

Functional Dependencies:

- `uid` → `email, full_name, phone, passport, password_hash, is_active, bookings_count, created_at, updated_at`
- `email` → `uid, full_name, phone, passport, password_hash, is_active, bookings_count, created_at, updated_at`
- `passport` → `uid, email, full_name, phone, password_hash, is_active, bookings_count, created_at, updated_at`

Candidate Keys: {uid}, {email}, {passport}

6.1.2 Destinations Table Functional Dependencies

Table Schema: `destinations(id, name, country, city, description, featured_image, is_active, created_at, updated_at)`

Functional Dependencies:

- `id` → `name, country, city, description, featured_image, is_active, created_at, updated_at`
- `(country, city)` → `name` (assuming unique city names per country)

Candidate Keys: {id}

6.1.3 Packages Table Functional Dependencies

Table Schema: `packages(id, title, description, price, destination_id, featured_image, is_featured, is_active, created_at, updated_at)`

Functional Dependencies:

- `id` → `title, description, price, destination_id, featured_image, is_featured, is_active, created_at, updated_at`
- `destination_id` $\nrightarrow$ `id` (Not a functional dependency — one destination can have multiple packages)

Candidate Keys: {id}

6.1.4 Package Detail Schedule Table Functional Dependencies

Table Schema: `package_detail_schedule(id, package_id, highlights, itinerary, duration_days, created_at, updated_at)`

Functional Dependencies:

- `id` → `package_id, highlights, itinerary, duration_days, created_at, updated_at`
- `package_id` → `id, highlights, itinerary, duration_days, created_at, updated_at` (One-to-one relationship)

Candidate Keys: {id}, {package_id}

6.1.5 Promo Codes Table Functional Dependencies

Table Schema: `promo_codes(id, code, description, discount_type, discount_value, start_date, expiry_date, is_active)`

Functional Dependencies:

- `id` → `code, description, discount_type, discount_value, start_date, expiry_date, is_active`
- `code` → `id, description, discount_type, discount_value, start_date, expiry_date, is_active`

Candidate Keys: {id}, {code}

6.1.6 Bookings Table Functional Dependencies

Table Schema: `bookings(id, package_id, user_id, promo_code_id, status, payment_status, booking_date, total_amount, paid_amount, discount_amount)`

Functional Dependencies:

- `id` → `package_id, user_id, promo_code_id, status, payment_status, booking_date, total_amount, paid_amount, discount_amount`
- `(user_id, package_id, booking_date)` → `id` (Composite candidate key in some business scenarios)

Candidate Keys: {id}

6.1.7 Blogs Table Functional Dependencies

Table Schema: `blogs(id, title, content, featured_image, author_id, is_published, created_at, updated_at)`

Functional Dependencies:

- `id` → `title, content, featured_image, author_id, is_published, created_at, updated_at`
- `title` → `id, content, featured_image, author_id, is_published, created_at, updated_at` (Assuming unique titles)

Candidate Keys: {id}, {title}

6.1.8 Admins Table Functional Dependencies

Table Schema: `admins(id, username, email, full_name, password_hash, is_super_admin, is_active, created_at, updated_at)`

Functional Dependencies:

- `id` → `username, email, full_name, password_hash, is_super_admin, is_active, created_at, updated_at`
- `username` → `id, email, full_name, password_hash, is_super_admin, is_active, created_at, updated_at`
- `email` → `id, username, full_name, password_hash, is_super_admin, is_active, created_at, updated_at`

Candidate Keys: {id}, {username}, {email}

6.1.9 Package Images Table Functional Dependencies

Table Schema: `package_images(id, package_id, image_url, alt_text, is_primary, created_at)`

Functional Dependencies:

- `id` → `package_id, image_url, alt_text, is_primary, created_at`
- `(package_id, is_primary)` → `id, image_url, alt_text, created_at` (Only one primary image per package)

Candidate Keys: {id}

6.2 Normalization Analysis

6.2.1 First Normal Form (1NF)

Definition: A table is in 1NF if:

- All attributes contain atomic (indivisible) values
- No repeating groups exist
- Each row is uniquely identifiable

Analysis - All Tables Compliance:

```
-- VIOLATES 1NF (Before normalization)
CREATE TABLE users_unnormalized (
    uid uuid,
    email varchar(255),
    full_name varchar(255),
    phone_numbers varchar(500), -- "123-456-7890, 987-654-3210"
    addresses text,             -- "123 Main St, City A; 456 Oak Ave,
   City B"
    skills text                 -- "Java, Python, JavaScript"
);

-- COMPLIES WITH 1NF (Current design)
CREATE TABLE users (
    uid uuid PRIMARY KEY,
    email varchar(255) UNIQUE NOT NULL,
    full_name varchar(255) NOT NULL,
    phone varchar(20),          -- Single atomic phone value
    passport varchar(255) UNIQUE NOT NULL,
    password_hash varchar(255) NOT NULL,
    is_active boolean NOT NULL DEFAULT true,
    bookings_count integer DEFAULT 0,
    created_at timestamp DEFAULT CURRENT_TIMESTAMP,
    updated_at timestamp DEFAULT CURRENT_TIMESTAMP
);
```

Listing 33: 1NF Compliance Example - Users Table

1NF Status: All nine tables comply with 1NF as each attribute contains atomic values.

6.2.2 Second Normal Form (2NF)

Definition: A table is in 2NF if:

- It is in 1NF
- No partial dependencies exist (non-key attributes depend on entire primary key)

Analysis - Composite Key Scenarios:

```
-- VIOLATES 2NF (Hypothetical composite key scenario)
CREATE TABLE package_images_violation (
    package_id uuid,
    image_sequence integer,
```

```
    package_title varchar(255),     -- Depends only on package_id (partial dependency)
    package_price numeric(10,2),    -- Depends only on package_id (partial dependency)
    image_url text,                 -- Depends on both package_id, image_sequence
    alt_text varchar(255),          -- Depends on both package_id, image_sequence
    is_primary boolean,             -- Depends on both package_id, image_sequence
    PRIMARY KEY (package_id, image_sequence)
);

-- COMPLIES WITH 2NF (Current design)
CREATE TABLE packages (
    id uuid PRIMARY KEY,
    title varchar(255) NOT NULL,    -- Depends only on package id
    price numeric(10,2) NOT NULL    -- Depends only on package id
);

CREATE TABLE package_images (
    id uuid PRIMARY KEY,            -- Single attribute primary key
    package_id uuid NOT NULL,       -- References packages.id
    image_url text NOT NULL,        -- Depends on image id
    alt_text varchar(255),          -- Depends on image id
    is_primary boolean DEFAULT false, -- Depends on image id
    FOREIGN KEY (package_id) REFERENCES packages(id)
);
```

Listing 34: 2NF Analysis - Package Images with Composite Keys

2NF Status: All tables comply with 2NF because we use single-attribute UUID primary keys, eliminating partial dependency issues.

6.2.3 Third Normal Form (3NF)

Definition: A table is in 3NF if:

- It is in 2NF
- No transitive dependencies exist (non-key attributes don't depend on other non-key attributes)

Analysis - Transitive Dependency Elimination:

```
-- VIOLATES 3NF (Before normalization)
CREATE TABLE bookings_unnormalized (
    id uuid PRIMARY KEY,
    package_id uuid,
    user_id uuid,
    package_title varchar(255),     -- Transitively depends on package_id
    package_price numeric(10,2),    -- Transitively depends on package_id
    destination_name varchar(255),  -- Transitively depends on package_id     destination_id
    user_email varchar(255),        -- Transitively depends on user_id
    user_full_name varchar(255),    -- Transitively depends on user_id
```

```
    promo_code varchar(50),          -- Transitively depends on
   promo_code_id
    discount_percentage double,      -- Transitively depends on
   promo_code_id
    booking_date timestamp,
    total_amount numeric(10,2),
    status varchar(20)
);

-- COMPLIES WITH 3NF (Current normalized design)
CREATE TABLE bookings (
    id uuid PRIMARY KEY,
    package_id uuid NOT NULL,        -- References packages.id
    user_id uuid NOT NULL,           -- References users.uid
    promo_code_id uuid,              -- References promo_codes.id
    status varchar(20) NOT NULL DEFAULT 'pending',
    payment_status varchar(20) NOT NULL DEFAULT 'pending',
    booking_date timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
    total_amount numeric(10,2) NOT NULL,
    paid_amount numeric(10,2) DEFAULT 0,
    discount_amount numeric(10,2) DEFAULT 0,
    FOREIGN KEY (package_id) REFERENCES packages(id),
    FOREIGN KEY (user_id) REFERENCES users(uid),
    FOREIGN KEY (promo_code_id) REFERENCES promo_codes(id)
);

-- Related information stored in separate normalized tables
CREATE TABLE packages (
    id uuid PRIMARY KEY,
    title varchar(255) NOT NULL,
    price numeric(10,2) NOT NULL,
    destination_id uuid NOT NULL,
    FOREIGN KEY (destination_id) REFERENCES destinations(id)
);

CREATE TABLE destinations (
    id uuid PRIMARY KEY,
    name varchar(255) NOT NULL,
    country varchar(100) NOT NULL,
    city varchar(100) NOT NULL
);

CREATE TABLE users (
    uid uuid PRIMARY KEY,
    email varchar(255) UNIQUE NOT NULL,
    full_name varchar(255) NOT NULL
);

CREATE TABLE promo_codes (
    id uuid PRIMARY KEY,
    code varchar(50) UNIQUE NOT NULL,
    discount_type varchar(20) NOT NULL,
    discount_value double precision NOT NULL
);
```

Listing 35: 3NF Analysis - Bookings Table

3NF Status: All tables achieve 3NF by storing related information in separate

tables and using foreign key references, eliminating transitive dependencies.

6.2.4 Boyce-Codd Normal Form (BCNF)

Definition: A relation is in BCNF if:

- It is in 3NF
- For every non-trivial functional dependency X → Y, X is a superkey

Superkey Definition: A superkey is a set of attributes that can uniquely identify each tuple in a relation. Every candidate key is a superkey, but not every superkey is a candidate key.

Analysis - BCNF Compliance for All Tables:

```
-- POTENTIAL BCNF VIOLATION (Hypothetical scenario)
CREATE TABLE blogs_violation (
    id uuid PRIMARY KEY,
    title varchar(255) UNIQUE,
    content text,
    author_username varchar(50),     -- Not a superkey
    author_email varchar(255),       -- Depends on author_username
    author_full_name varchar(255),   -- Depends on author_username
    is_published boolean
);
-- Here: author_username     author_email, author_full_name (violates
   BCNF)
-- author_username is not a superkey since it cannot uniquely identify
   tuples

-- COMPLIES WITH BCNF (Current design)
CREATE TABLE blogs (
    id uuid PRIMARY KEY,
    title varchar(255) UNIQUE NOT NULL,
    content text NOT NULL,
    featured_image text,
    author_id uuid NOT NULL,         -- References admins.id (superkey)
    is_published boolean DEFAULT false,
    created_at timestamp DEFAULT CURRENT_TIMESTAMP,
    updated_at timestamp DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (author_id) REFERENCES admins(id)
);

CREATE TABLE admins (
    id uuid PRIMARY KEY,             -- Superkey
    username varchar(50) UNIQUE,     -- Superkey
    email varchar(255) UNIQUE,       -- Superkey
    full_name varchar(255) NOT NULL,
    password_hash varchar(255) NOT NULL,
    is_super_admin boolean DEFAULT false,
    is_active boolean DEFAULT true
);
```

Listing 36: BCNF Analysis - Superkey-Based Review

BCNF Compliance Summary (Superkey-Based Analysis):

1. **Users Table:** BCNF Compliant

- **Superkeys:** {uid}, {email}, {passport}, and their supersets
- All non-trivial functional dependencies have a superkey as the determinant
- Example: uid → full_name, email, phone (uid is a superkey)

2. **Destinations Table:** BCNF Compliant
   - **Superkey:** {id}
   - All attributes depend only on the superkey
   - Example: id → name, country, city (id is a superkey)
3. **Packages Table:** BCNF Compliant
   - **Superkey:** {id}
   - No non-superkey determines any other attribute
   - Example: id → title, price, destination_id (id is a superkey)
4. **Package Detail Schedule Table:** BCNF Compliant
   - **Superkeys:** {id}, {package_id} (if one-to-one relationship)
   - All dependencies are on superkeys; no partial or transitive dependency
   - Example: id → highlights, itinerary (id is a superkey)
5. **Promo Codes Table:** BCNF Compliant
   - **Superkeys:** {id}, {code}
   - Every functional dependency has a superkey as the determinant
   - Example: code → discount_type, discount_value (code is a superkey)
6. **Bookings Table:** BCNF Compliant
   - **Superkey:** {id}
   - All attributes are functionally dependent only on a superkey
   - Example: id → user_id, package_id, total_amount (id is a superkey)
7. **Blogs Table:** BCNF Compliant
   - **Superkeys:** {id}, {title}
   - All dependencies originate from superkeys
   - Example: title → content, author_id (title is a superkey)
8. **Admins Table:** BCNF Compliant
   - **Superkeys:** {id}, {username}, {email}
   - No functional dependency violates the BCNF condition
   - Example: username → full_name, password_hash (username is a superkey)
9. **Package Images Table:** BCNF Compliant
   - **Superkey:** {id}
   - Foreign key to package is not a violating dependency; id determines all
   - Example: id → package_id, image_url, alt_text (id is a superkey)

7 Frontend Design and Implementation

7.1 Frontend Architecture

7.1.1 Technology Stack

The frontend implements a modern technology stack:

- **Framework:** Next.js 15 with App Router
- **Language:** TypeScript for type safety
- **Styling:** Tailwind CSS 4 with custom design system
- **UI Components:** ShadCN UI with Radix UI primitives
- **State Management:** Zustand for global state
- **API Integration:** Custom API client with error handling
- **Authentication:** JWT with role-based access control

7.2 Frontend Architecture Advantages

7.2.1 Advantages

- **Type Safety:** TypeScript prevents runtime errors
- **Performance:** Next.js optimizations and code splitting
- **SEO:** Server-side rendering capabilities
- **Maintainability:** Component-based architecture
- **User Experience:** Responsive and intuitive interface
- **Developer Experience:** Hot reload and modern tooling

7.2.2 Disadvantages

- **Complexity:** Steep learning curve for modern stack
- **Bundle Size:** React and Next.js add overhead
- **Build Time:** TypeScript compilation increases build duration
- **Dependency Management:** Multiple dependencies require maintenance

7.3 Key Features Implementation

7.3.1 User Management

- Secure registration and login processes
- Profile management with real-time updates
- Password reset and account verification
- Role-based dashboard access

7.3.2 Package Management

- Advanced search and filtering capabilities
- Interactive package browsing with image galleries
- Real-time availability and pricing updates
- Responsive card-based layout

7.3.3 Booking System

- Multi-step booking process with validation
- Promo code integration and validation
- Payment status tracking and notifications
- Booking history and management

7.3.4 Admin Dashboard

- Comprehensive analytics and reporting
- CRUD operations for all entities
- Real-time data visualization
- User and content management tools

7.4 API Integration

7.4.1 API Endpoints

The backend exposes several RESTful API endpoints for frontend integration:

Endpoint	Method	Description
/api/auth/login	`POST`	User authentication and JWT token generation
/api/auth/register	`POST`	New user registration with validation

Endpoint	Method	Description
/api/auth/refresh	POST	JWT token refresh for session management
/api/packages	GET	Retrieve all active travel packages
/api/packages/{id}	GET	Get specific package details by ID
/api/packages/search	POST	Advanced package search with filters
/api/packages/featured	GET	Get featured travel packages
/api/destinations	GET	List all available destinations
/api/destinations/{id}	GET	Get destination details and packages
/api/destinations/popular	GET	Retrieve popular destinations
/api/bookings	GET	User booking history and status
/api/bookings	POST	Create new booking with validation
/api/bookings/{id}	PUT	Update booking status or details
/api/bookings/{id}/cancel	POST	Cancel existing booking
/api/users/profile	GET	Get user profile information
/api/users/profile	PUT	Update user profile data
/api/users/bookings	GET	User-specific booking history
/api/promo-codes/validate	POST	Validate promotional codes
/api/blogs	GET	Retrieve travel blogs and articles
/api/blogs/{id}	GET	Get specific blog content
/api/admin/dashboard	GET	Administrative analytics data
/api/admin/packages	POST, PUT, DELETE	Package management operations
/api/admin/users	GET	User management and statistics
/api/admin/bookings	GET	Booking management and reports

7.5 Website Snapshots

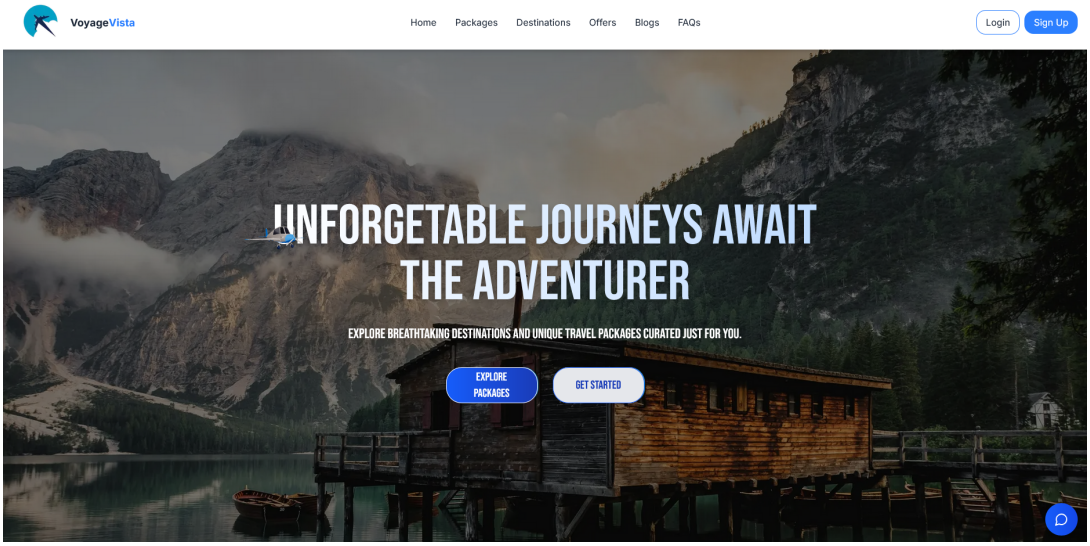

Figure 3: Landing Page

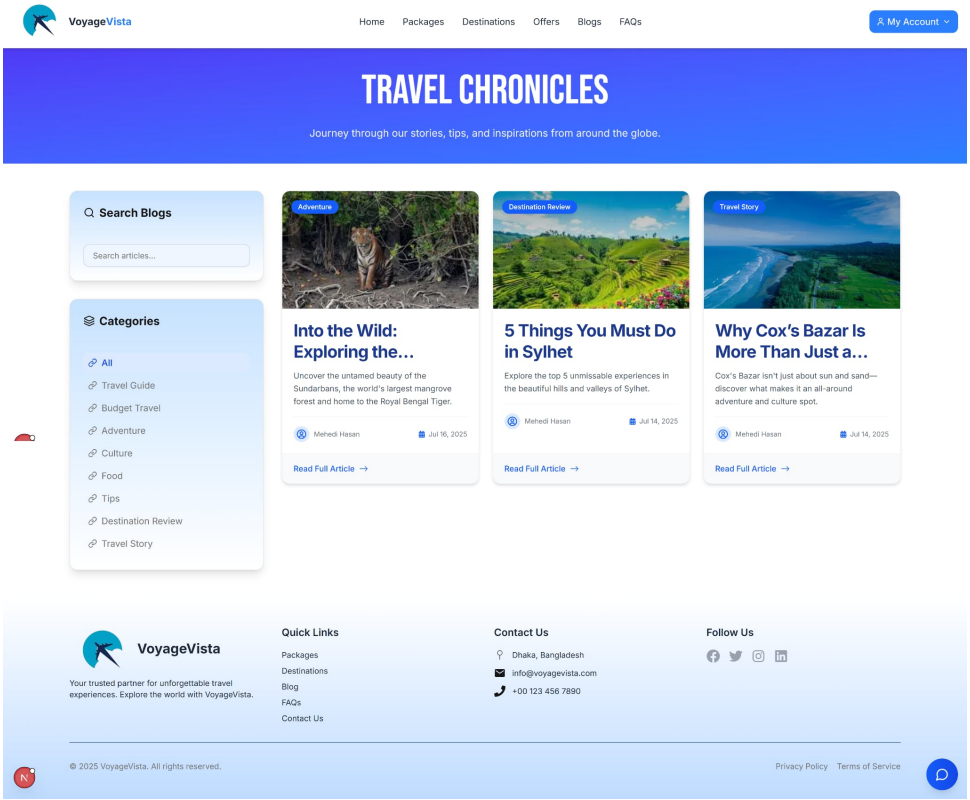

Figure 4: Blogs Page

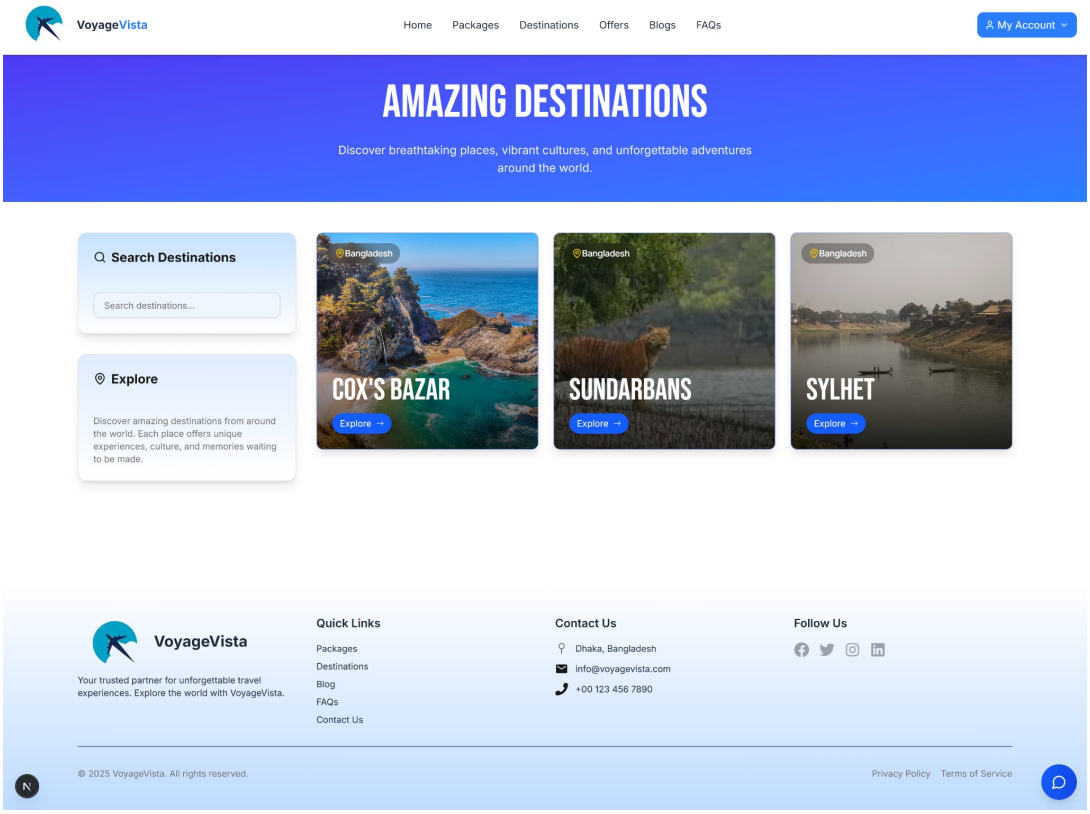

Figure 5: Destinations Page

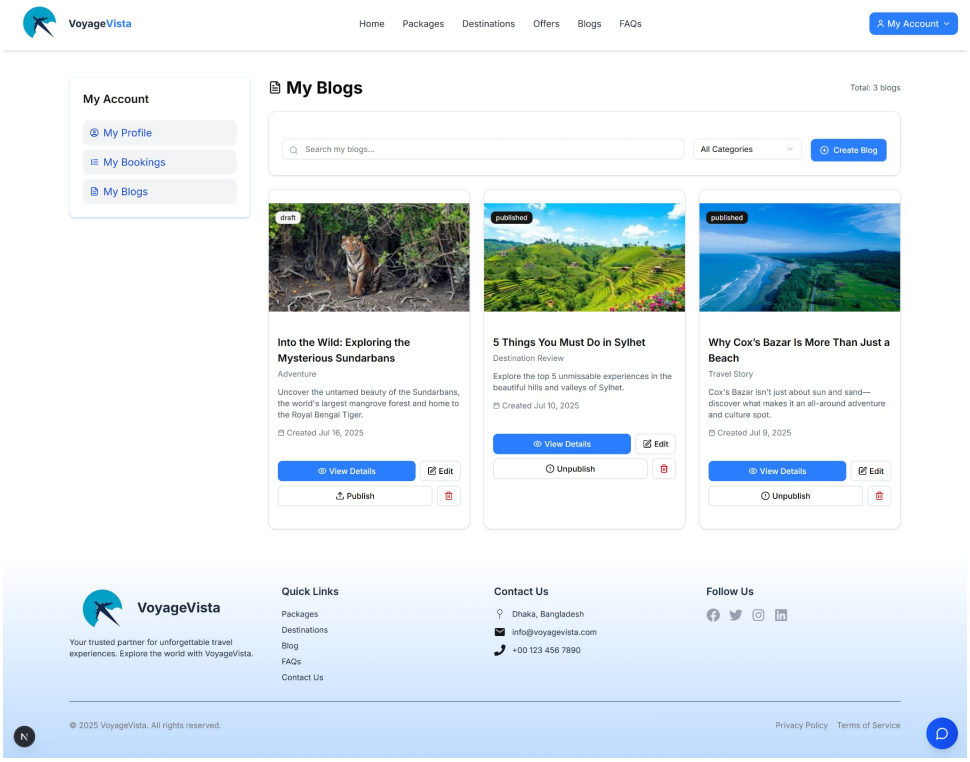

Figure 6: My Blogs Page

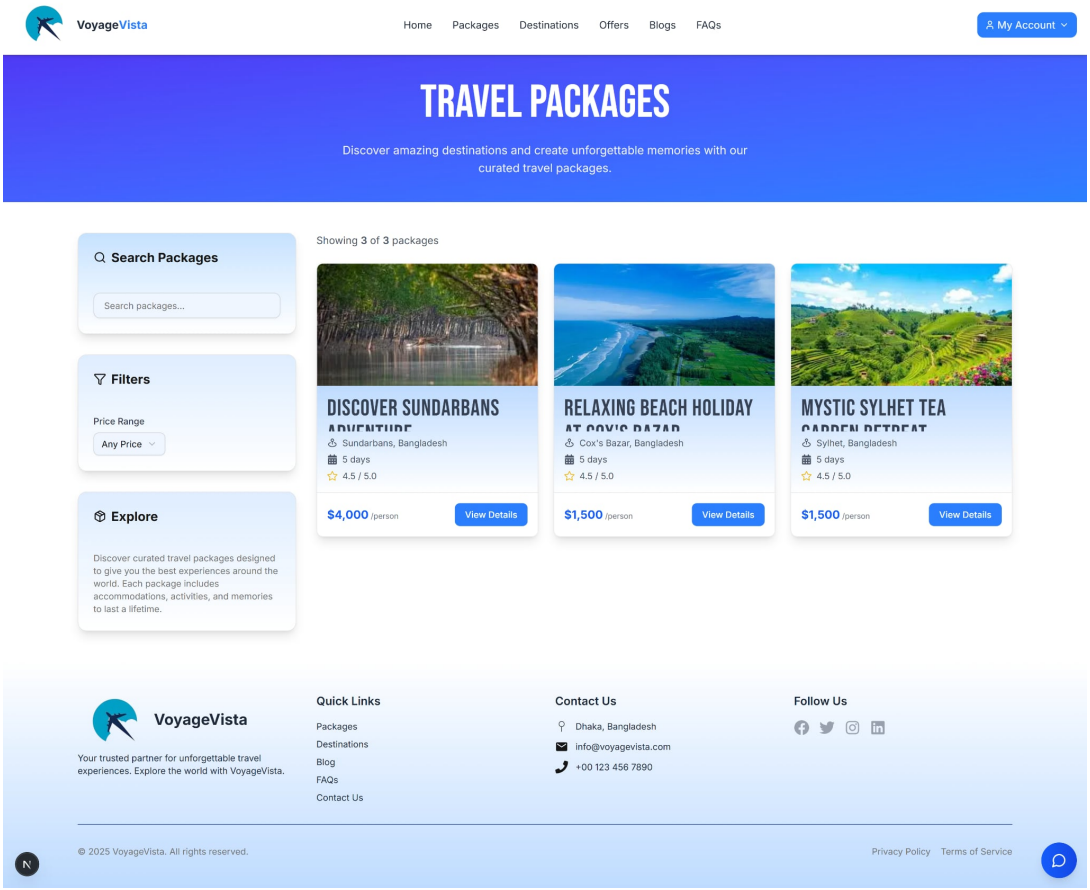

Figure 7: Packages Page

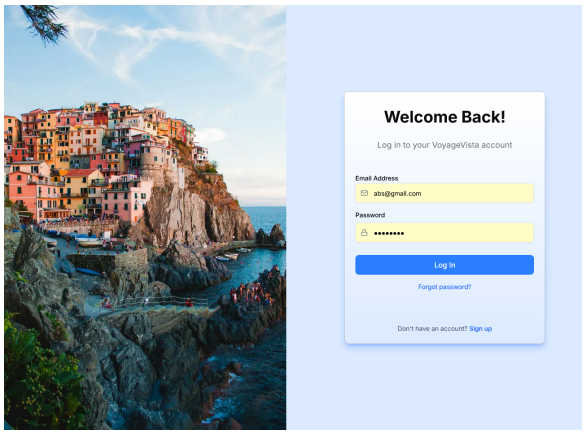

Figure 8: Login Page

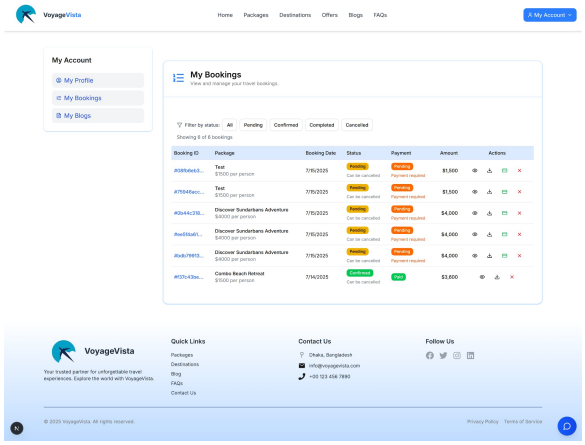

Figure 9: My Bookings

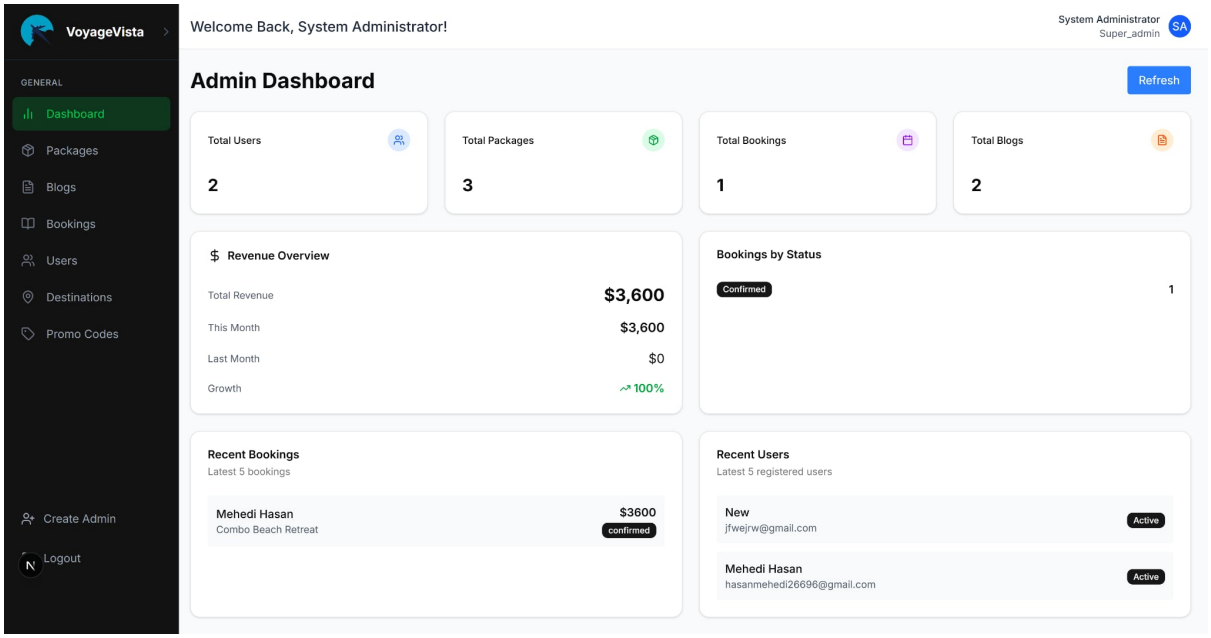

Figure 10: Admin Dashboard Page

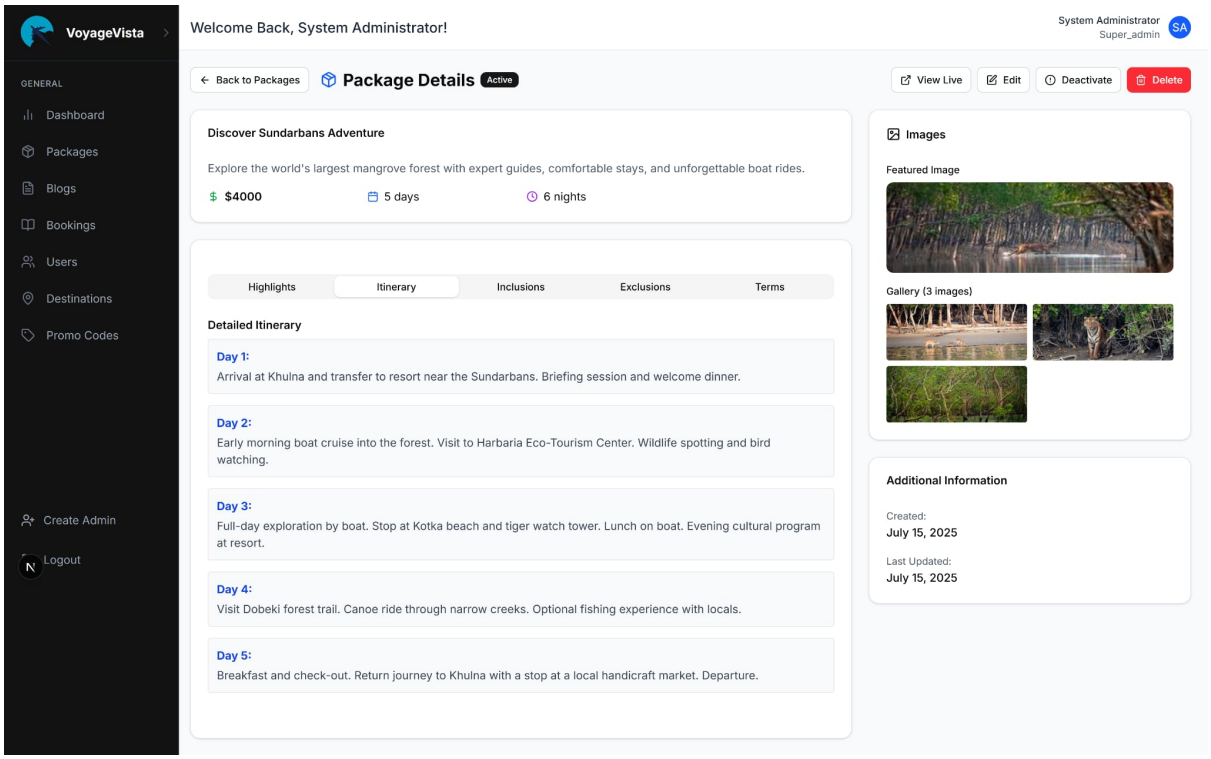

Figure 11: Admin Package Details Page

8 Conclusion

8.1 Summary

VoyageVista presents a modern, end-to-end travel booking platform that integrates a well-structured relational database with a contemporary web application. The system effectively demonstrates key concepts in database design, SQL optimization, secure authentication, and responsive frontend development. It offers a scalable and user-friendly solution that meets real-world travel management needs.

8.2 Learning Outcomes

Through this project, we gained valuable experience in:

- Database design principles and normalization techniques
- Complex SQL query formulation and optimization
- Modern web development frameworks and technologies
- System architecture and design patterns
- Frontend-backend integration and API design
- Security implementation and best practices

8.3 Future Enhancements

Potential improvements for the system include:

- Implementation of advanced analytics and machine learning
- Mobile application development
- Payment gateway integration
- Real-time notifications and communication
- Advanced search with AI-powered recommendations
- Multi-language support and internationalization