# Database Management Systems (IT214)
# Tourism Management
# (Team ID - T209)

| NAMES | IDs |
|---|---|
| SAHILKUMAR SUTARIYA | 202201119 |
| MEET ANDHARIA | 202201145 |
| DEV DODIYA | 202201153 |
| MALHAR VAGHASIYA | 202201183 |
| AKSHAT JOSHI | 202201185 |

**Contact No. of Group Representative: 7861881819**
**(Sahilkumar Sutariya, 202201119)**

**Prof. P. M. Jat**
**TA: Sagar Joshi**

# SUMMARY

The primary objective of our tourism management database project is to streamline the management of travel-related information, enhancing the user experience through efficient organization and utilization of data. Initially, we categorize users into two groups: logged-in customers and non-logged-in users, assigning separate tables for each. We provide a variety of destinations along with their transportation, accommodation, and activities, tracking their details within the system. Customers' preferences including destination, accommodation, and activities, are categorized to tailor personalized experiences. Upon booking, essential details of the bookings are stored in the system. Separate tables for accommodations, transportation, and activities facilitate comprehensive tour planning and management, enabling tour operators (admins) to efficiently handle their customers' travel data. Admins can keep track of the customer details who have booked their service. For financial management, we collect data on tour expenses such as accommodation, transportation, and activity charges, revenues, and payments, ensuring accurate financial reporting. This comprehensive approach ensures the development of an efficient database system for tourism management.

# EXPERIENCE

During the creation of our project, Tourism Management, we faced several challenges along the way. One significant hurdle that we faced was during the development of the Entity-Relationship (ER) diagram. We struggled to determine the key attributes of each entity and establish their relationships with others.

Even after finalizing the ER diagram, we faced difficulties in creating queries and implementing them in SQL.

Additionally, when we learned about normalization, we realized the necessity of updating our relational schema, ER diagram, and DDL scripts accordingly.

Even though we faced some tough challenges, working through them made us work better together and understand how databases work even more.

From this project, we learned that it's important to keep trying even when things get tough. Working together helped us solve problems better, and we realized the need to keep learning about new concepts like normalization. We also got better at managing complex databases, solving problems and working as a team. This experience taught us a lot that we can use in the future.

# TOP-3 QUERIES

1. **Retrieve customers who have booked accommodations in destinations where the best month to visit is in the summer (June, July, August) and have left a review with a rating above 4.**

   ```
   CREATE VIEW HighRatedSummerBookings AS
   SELECT c.Fname, c.Lname, r.Rating, d.Dname
   FROM Customer c
   JOIN Booking b ON c.CustId = b.CustID
   JOIN Destination d ON b.DID = d.DID
   JOIN Review r ON c.CustId = r.CustID AND b.BookingID = r.BookingID
   WHERE EXTRACT(MONTH FROM b.Checkin_Date) IN (6, 7, 8) AND
   r.Rating > 4;

   SELECT * FROM HighRatedSummerBookings;
   ```

2. **List top 5 destinations which are popular among the customers which fall in the age group of 18-30.**

   ```
   select distinct d.DID, DName, Country from
   Destination as d natural join Booking as b
   natural join Customer as c
   where DOB between '1994-01-01' and '2006-01-01'
   limit 5;
   ```

3. **Identify top 3 destinations where the number of bookings exceeds the average number of bookings.**

   ```
   SELECT D.DID, D.Dname, COUNT(B.BookingID) AS Total_Bookings
   FROM Destination D
   LEFT JOIN Booking B ON D.DID = B.DID
   GROUP BY D.DID, D.Dname
   HAVING COUNT(B.BookingID) > (SELECT AVG(Booking_Count)
   FROM
   (SELECT COUNT(BookingID) AS Booking_Count FROM Booking
   GROUP BY DID) AS AvgBookings) Limit 3;
   ```

# DDL SCRIPT

```
create table Customer(
        CustId INT PRIMARY KEY,
        Password VARCHAR(20) NOT NULL,
        Fname VARCHAR(20) NOT NULL,
        Mname VARCHAR(20),
        Lname VARCHAR(20) NOT NULL,
        DOB DATE
);

create table Customer_Contact(
        Contact BIGINT,
        CustID INT,
        PRIMARY KEY (Contact, CustID),
        FOREIGN KEY (CustID) REFERENCES Customer(CustID)
        ON DELETE CASCADE
);

create table Customer_Email(
        Email VARCHAR(60),
        CustID INT,
        PRIMARY KEY (Email, CustID),
        FOREIGN KEY (CustID) REFERENCES Customer(CustID)
        ON DELETE CASCADE
);

create table Emergency_Contact(
        Contact_Name VARCHAR(40),
        CustID INT,
        Address VARCHAR(100),
        Relation VARCHAR(20),
        Contact_Number BIGINT NOT NULL,
        PRIMARY KEY (Contact_Name, CustID),
        FOREIGN KEY (CustID) REFERENCES Customer(CustID)
        ON DELETE CASCADE
);
```

```
create table Destination(
        DID INT PRIMARY KEY,
        Dname VARCHAR(30),
        Country VARCHAR(30),
        Description VARCHAR(100),
        Best_month_to_visit VARCHAR(10)
);

create table Popular_Attractions(
        Popular_Attractions VARCHAR(100),
        DID INT,
        PRIMARY KEY (Popular_Attractions, DID),
        FOREIGN KEY (DID) REFERENCES Destination(DID)
        ON DELETE CASCADE
);

create table Non_logged_in_user(
        User_Name VARCHAR(30),
        DID INT,
        PRIMARY KEY (User_Name, DID),
        FOREIGN KEY (DID) REFERENCES Destination(DID)
        ON DELETE CASCADE
);


create table Admin(
        AdminID INT PRIMARY KEY,
        Type VARCHAR(20) NOT NULL,
        Fname VARCHAR(20) NOT NULL,
        Mname VARCHAR(20),
        Lname VARCHAR(20) NOT NULL
);

create table Booking(
        BookingID BIGINT PRIMARY KEY,
        Booking_Date DATE NOT NULL,
        Booking_Status VARCHAR(20) NOT NULL,
        Booking_Type VARCHAR(30) NOT NULL,
        Total_Cost INT NOT NULL,
        No_of_Guests INT NOT NULL,
        Checkin_Date  DATE NOT NULL,
        Checkout_Date DATE NOT NULL,
        CustID INT,
        DID INT,
        AdminID INT,
        FOREIGN KEY (CustID) REFERENCES Customer(CustID)
        ON DELETE CASCADE,
        FOREIGN KEY (DID) REFERENCES Destination(DID)
        ON DELETE CASCADE,
        FOREIGN KEY (AdminID) REFERENCES Admin(AdminID)
        ON DELETE CASCADE
);

create table Payment(
        TransactionID BIGINT PRIMARY KEY,
        Method VARCHAR(20) NOT NULL,
        Amount INT NOT NULL,
        Transaction_Date DATE NOT NULL,
        Transaction_Status VARCHAR(20) NOT NULL,
        CustID INT,
        DID INT,
        AdminID INT,
        BookingID INT,
        FOREIGN KEY (CustID) REFERENCES Customer(CustID)
        ON DELETE CASCADE,
        FOREIGN KEY (DID) REFERENCES Destination(DID)
        ON DELETE CASCADE,
        FOREIGN KEY (AdminID) REFERENCES Admin(AdminID)
        ON DELETE CASCADE,
        FOREIGN KEY (BookingID) REFERENCES Booking(BookingID)
        ON DELETE CASCADE
);
```

```
create table Review(
        R_Type VARCHAR(20),
        Rating FLOAT NOT NULL,
        CustID INT,
        BookingID BIGINT,
        Text VARCHAR(100),
        R_Date DATE NOT NULL,
        PRIMARY KEY (R_Type, CustID, BookingID),
        FOREIGN KEY (CustID) REFERENCES Customer(CustID)
        ON DELETE CASCADE,
        FOREIGN KEY (BookingID) REFERENCES Booking(BookingID)
        ON DELETE CASCADE
);

create table AdminEmail(
        Email VARCHAR(60),
        AdminID INT,
        PRIMARY KEY (AdminID,Email),
        FOREIGN KEY (AdminID) REFERENCES Admin(AdminID)
        ON DELETE CASCADE
);

create table AdminContact(
        Contact BIGINT,
        AdminID INT,
        PRIMARY KEY (AdminID,Contact),
        FOREIGN KEY (AdminID) REFERENCES Admin(AdminID)
        ON DELETE CASCADE
);

create table EventsAndFests(
        Event_Name VARCHAR(100),
        Event_Date DATE,
        Description VARCHAR(100),
        DID INT,
        PRIMARY KEY (Event_Name,Event_Date,DID),
        FOREIGN KEY (DID) REFERENCES Destination(DID)
```

```
    ON DELETE CASCADE
);

create table Transportation(
    Transportation_ID INT PRIMARY KEY,
    Price INT NOT NULL,
    Capacity INT NOT NULL,
    Trans_Type VARCHAR(20) NOT NULL,
    Trans_Name VARCHAR(40) NOT NULL,
    DID INT,
    AdminID INT,
    FOREIGN KEY (DID) REFERENCES Destination(DID)
    ON DELETE CASCADE,
    FOREIGN KEY (AdminID) REFERENCES Admin(AdminID)
    ON DELETE CASCADE
);

create table Accommodation(
    AccommodationID INT PRIMARY KEY,
    Aname VARCHAR(40) NOT NULL,
    Atype VARCHAR(20) NOT NULL,
    Price_per_night INT NOT NULL,
    Availability INT NOT NULL,
    Docs_Required VARCHAR(100),
    DID INT,
    AdminID INT,
    FOREIGN KEY (DID) REFERENCES Destination(DID)
    ON DELETE CASCADE,
    FOREIGN KEY (AdminID) REFERENCES Admin(AdminID)
    ON DELETE CASCADE
);

create table Activities(
    AcivityID INT PRIMARY KEY,
    Activity_type VARCHAR(20) NOT NULL,
    Activity_name VARCHAR(100) NOT NULL,
    Price INT NOT NULL,
    Availability INT,
    DID INT,
    AdminID INT,
    FOREIGN KEY (DID) REFERENCES Destination(DID)
    ON DELETE CASCADE,
    FOREIGN KEY (AdminID) REFERENCES Admin(AdminID)
    ON DELETE CASCADE
);

create table Refund(
    Refund_Status VARCHAR(20),
    BookingID BIGINT,
    TransactionID BIGINT,
    PRIMARY KEY (Refund_Status, BookingID, TransactionID),
    FOREIGN KEY (BookingID) REFERENCES Booking(BookingID)
    ON DELETE CASCADE,
    FOREIGN KEY (TransactionID) REFERENCES Payment(TransactionID)
    ON DELETE CASCADE
);
```

# SQL QUERIES

**CUSTOMERS :-**

1. **Retrieve customers who have booked accommodations in destinations where the best month to visit is in the summer (June, July, August) and have left a review with a rating above 4**

```
CREATE VIEW HighRatedSummerBookings AS
SELECT c.Fname, c.Lname, r.Rating, d.Dname
FROM Customer c
JOIN Booking b ON c.CustId = b.CustID
JOIN Destination d ON b.DID = d.DID
JOIN Review r ON c.CustId = r.CustID AND b.BookingID =
r.BookingID
WHERE EXTRACT(MONTH FROM b.Checkin_Date) IN (6, 7, 8)
AND r.Rating > 4;

SELECT * FROM HighRatedSummerBookings;
```

2. **List of all bookings of a particular customer having customerID 19 made in the year 2024.**

```
select BookingID, Total_Cost, Booking_date
from Booking
where CustID = 19
and Booking_date between '2024-01-01' and '2024-12-31';
```

3. **Retrieve destination name and country where the avg rating of activities is above 4.5**

```
SELECT d.dname,d.country
FROM Activities as a
JOIN Destination as d ON a.DID = d.DID
JOIN Booking as b ON d.DID = b.DID
JOIN Review as r ON r.bookingId=b.bookingId
GROUP BY d.DID
```

```
HAVING AVG(r.Rating) > 4.5;
```

4. **Retrieve refund status for cancelled bookings**
   ```
   SELECT b.BookingID, r.Refund_Status
   FROM Booking b
   JOIN Refund r ON b.BookingID = r.BookingID
   WHERE b.Booking_Status = 'Cancelled';
   ```

5. **Give list of accommodations at a specific location**
   ```
   select a.Aname, a.Atype, d.dname, a.Price_per_night, a.Availability,
   a.Docs_required
   from Accommodation as a natural join
   Destination as d
   where d.Country = 'France'
   order by a.Price_per_night desc;
   ```

**<u>ADMIN</u> :-**

1. **Retrieve all admins who are responsible for managing a specific destination (e.g., New York)**
   ```
   SELECT a.Fname, a.Lname, ac.Contact, d.aname, d.atype,
   de.dname
   FROM Admin a
   JOIN AdminContact ac ON a.AdminID = ac.AdminID
   JOIN Accommodation d ON a.AdminID = d.AdminID
   JOIN Destination de ON  d.DID = de.DID
   WHERE d.DID = 205 LIMIT 2;
   ```

2. **List of all Customers and their Booking Date who opted for UPI mode of payment in the booking process.**
   ```
   select p.CustId, b.Booking_Date from
   Payment as p join Booking as b
   on p.BookingId = b.BookingId
   ```

```
where p.method = 'UPI';
```

3. **Provide list of TransportationIDs booked by customers owned by Admin having ID = 1011 in month of may 2024.**

```
select Transportation_ID, Trans_Name, Trans_Type, Booking_date,
b.Checkin_Date
from Booking as b join Transportation as tr
on b.AdminID = tr.AdminID
where b.AdminID = 1011
and Checkin_Date between '2024-05-01' and '2024-05-31';
```

4. **Retrieve income of an admin for year 2024**

```
select a.AdminId, a.Fname, a.Lname, sum(total_cost) as income
from Booking as b natural join Admin as a
where b.AdminId = '1019' and
b.Booking_Date between '2024-01-01' and '2024-12-31'
group by a.AdminId;
```

5. **Retrieve emergency contact details of a particular customer in case of an emergency**

```
select c.fname, c.lname, e.* from
Customer as c natural join Emergency_Contact as e
where c.CustId = '10';
```

6. **Identify top 3 destinations where the number of bookings exceeds the average number of bookings**

```
SELECT D.DID, D.Dname, COUNT(B.BookingID) AS Total_Bookings
FROM Destination D
LEFT JOIN Booking B ON D.DID = B.DID
GROUP BY D.DID, D.Dname
HAVING COUNT(B.BookingID) > (SELECT AVG(Booking_Count)
FROM
(SELECT COUNT(BookingID) AS Booking_Count FROM Booking
GROUP BY DID) AS AvgBookings) Limit 3;
```

## NON-LOGGED IN USERS :-

1. **Give list of Activities available at Goa in the increasing order of price.**

   ```
   select a.* from Activities as a
   natural join Destination as d
   where d.Dname = 'Goa'
   order by Price;
   ```

2. **Give list of transportations available at Goa having capacity of 2 persons in the ascending order of price.**

   ```
   select t.Transportation_ID, t.Trans_Name, t.Trans_Type, t.Price,
   t.Capacity
   from Transportation as t natural join Destination as d
   where d.Dname = 'Goa'
   and t.Capacity = 2
   order by t.Price;
   ```

3. **Retrieve the popular attractions for a given Country**

   ```
   SELECT pa.Popular_Attractions, d.DID, d.Dname
   FROM Destination d
   JOIN Popular_Attractions pa ON d.DID = pa.DID
   GROUP BY d.DID, d.Dname, pa.Popular_Attractions, d.Country
   HAVING d.Country = 'India';
   ```

4. **List top 5 destinations which are popular among the customers which fall in the age group of 18-30**

   ```
   select distinct d.DID, DName, Country from
   Destination as d natural join Booking as b
   natural join Customer as c
   where DOB between '1994-01-01' and '2006-01-01'
   limit 5;
   ```

5. **Retrieve Events and fests at a particular Country between given date interval**

```
select e.*, d.Dname from EventsAndFests as e
join Destination as d
on e.DID = d.DID
where d.Country = 'India'
and e.event_date between '2024-04-01' and '2024-07-31';
```

# NORMALIZATION PROOFS

## 1. 'Customer' Relation:

Attributes: {CustID, Password, Fname, Mname, Lname, DOB}

Functional Dependencies:

CustID → Password

CustID → Fname

CustID → Mname

CustID → Lname

CustID → DOB

Let X = CustID
X+ = {CustID, Password, Fname, Mname, Lname, DOB}
Thus, **Primary key = CustID**

The left side of all the FDs in the minimal set of FDs for the relation 'Customer' is CustID, which is the primary key of this relation, so "Customer" is in **BCNF**.

## 2. 'Emergency_Contact' Relation:

Attributes: {Contact_Name, CustID, Address, Relation, Contact_Number}

Functional Dependencies :

{ Contact_Name, CustID } → Address

{ Contact_Name, CustID } → Relation

{ Contact_Name, CustID } → Contact_Number

Let X = { Contact_Name, CustID }
X+ = { Contact_Name, CustID, Address, Relation, Contact_Number }
Thus, **Primary key = { Contact_Name, CustID }**

The left side of all the FDs in the minimal set of FDs for the relation ‘Emergency_Contact’ is { Contact_Name, CustID }, which is the primary key of this relation, so “Emergency_Contact” is in **BCNF**.

3. **‘Customer_Contact’ Relation:**

Attributes: {CustID, Contact}

**Primary key = {CustID, Contact}**

There are no Functional Dependencies in this relation as the only two attributes are CustID and Contact, which itself are the primary key.

Thus, the relation “Customer_Contact” is in **BCNF**.

4. **‘Customer_Email’ Relation:**

Attributes: {CustID, Email}

**Primary key = {CustID, Email}**

There are no Functional Dependencies in this relation as the only two attributes are CustID and Email, which itself are the primary key.

Thus, the relation “Customer_Email” is in **BCNF**.

5. **‘Review’ Relation:**

Attributes: { CustID, R_Type, Rating, Text, R_Date }

Functional Dependencies :

{ CustID, R_type } → Rating

{ CustID, R_type } → Text

{ CustID, R_type } → R_Date

Let X = { CustID, R_type }
X+ = { CustID, R_Type, Rating, Text, R_Date }
Thus, **Primary key = { CustID, R_type }**

The left side of all the FDs in the minimal set of FDs for the relation 'Review' is { CustID, R_type }, which is the primary key of this relation, so "Review" is in **<u>BCNF</u>**.

6. **'Payment' Relation:**

Attributes: { TransactionID, Method, Amount, Transaction_Date, Transaction_Status, CustID, BookingID, DID, AdminID }

Functional Dependencies:

TransactionID → Method

TransactionID → Amount

TransactionID → Transaction_Date

TransactionID → Transaction_Status

TransactionID → CustID

TransactionID → BookingID

TransactionID → DID

TransactionID → AdminID

Let X = TransactionID
X+ = { TransactionID, Method, Amount, Transaction_Date, Transaction_Status, CustID, BookingID, DID, AdminID }
Thus, **Primary key = TransactionID**

The left side of all the FDs in the minimal set of FDs for the relation 'Payment' is TransactionID, which is the primary key of this relation, so "Payment" is in **BCNF**.

**7. 'Booking' Relation:**

Attributes : { BookingID, Booking_Date, Booking_Status, Total_Cost, No_of_Guests, Checkin_Date, Checkout_Date, CustID, DID, AdminID }

Functional Dependencies :

BookingID → Booking_Date

BookingID → Booking_Status

BookingID → Total_Cost

BookingID → No_of_Guests

BookingID → Checkin_Date

BookingID → Checkout_Date

BookingID → CustID

BookingID → DID

BookingID → AdminID

Let X = BookingID
X+ = { BookingID, Booking_Date, Booking_Status, Total_Cost, No_of_Guests, Checkin_Date, Checkout_Date, CustID, DID, AdminID }
Thus, the **Primary key = BookingID**

The left side of all the FDs in the minimal set of FDs for the relation 'Booking' is BookingID, which is the primary key of this relation, so "Booking" is in **BCNF**.

**8. 'Admin' Relation:**

Attributes : { AdminID, Password, Type, Fname, Mname, Lname }

Functional Dependencies :

AdminID → Password

AdminID → Type

AdminID → Fname

AdminID → Mname

AdminID → Lname

Let X = AdminID
X+ = { AdminID, Password, Type, Fname, Mname, Lname }
Thus, **Primary key = AdminID**

The left side of all the FDs in the minimal set of FDs for the relation 'Admin' is AdminID, which is the primary key of this relation, so "Admin" is in **BCNF**.

### 9. 'Admin_Contact' Relation:

Attributes : { AdminID, Contact }

**Primary key = { AdminID, Contact }**

There are no Functional Dependencies in this relation as the only two attributes are AdminID and  Contact, which itself are the primary keys.

Thus the relation "Admin_Contact" is in **BCNF**.

### 10. 'Admin_Email' Relation:

Attributes : { AdminID, Email }

**Primary key = { AdminID, Email }**

There are no Functional Dependencies in this relation as the only two attributes are AdminID and  Email, which itself are the primary key.

Thus the relation “Admin_Email” is in **BCNF**.

### 11. ‘Non_logged_in_user’ Relation:

Attributes : { User_Name, DID }

**Primary key = { User_Name, DID }**

There are no Functional Dependencies in this relation as the only two attributes are User_Name and  DID, which itself are the primary key.

Thus the relation “Non_logged_in_user” is in **BCNF**.

### 12. ‘EventsAndFests’ Relation:

Attributes : { DID, Event_Name, Event_Date, Description }

Functional Dependencies :

{ DID, Event_Name, Event_Date } → Description

Let X = { DID, Event_Name, Event_Date }<br>
X+ = { DID, Event_Name, Event_Date, Description }<br>
Thus, **Primary key = { DID, Event_Name, Event_Date }**

The left side of all the FDs in the minimal set of FDs for the relation ‘EventsAndFests’ is { DID, Event_Name, Event_Date }, which is the primary key of this relation, so “EventsAndFests” is in **BCNF**.

### 13. ‘Popular_Attractions’ Relation:

Attributes : { Popular_Attractions, DID }

**Primary key = { Popular_Attractions, DID }**

There are no Functional Dependencies in this relation as the only two attributes are Popular_Attractions and  DID, which itself are the primary key.

Thus the relation “Popular_Attractions” is in **BCNF**.

### 14. ‘Destination’ Relation:

Attributes: { DID, Dname, Country, Description, Best_month_to_visit}

Functional Dependencies :

DID → Dname

DID → Country

DID → Description

DID → Best_month_to_visit

Let X = DID
X+ = { DID, Dname, Country, Description, Best_month_to_visit }
Thus, **Primary key = DID**

The left side of all the FDs in the minimal set of FDs for the relation ‘Destination’ is DID, which is the primary key of this relation, so “Destination” is in **BCNF**.

### 15. ‘Transportation’ Relation:

Attributes : { TransportationID, Price, Capacity, Trans_Type, DID, AdminID }

Functional Dependencies:

TransportationID → Price

TransportationID → Capacity

TransportationID → Trans_Type

TransportationID → DID

TransportationID → AdminID

Let X = TransportationID
X+ = {TransportationID, Price, Capacity, Trans_Type, DID, AdminID}
Thus, the **Primary key = TransportationID**

The left side of all the FDs in the minimal set of FDs for the relation ‘Transportation’ is TransportationID, which is the primary key of this relation, so “Transportation” is in **BCNF**.

### 16. ‘Accommodation’ Relation:

Attributes: {AccommodationID, Atype, Price_per_night, Aname, Availability, Docs_Required, DID, AdminID}

Functional Dependencies:

AccommodationID → Atype

AccommodationID → Price_per_night

AccommodationID → Anime

AccommodationID → Availability

AccommodationID → Docs_Required

AccommodationID → DID

AccommodationID → AdminID

Let X = AccommodationID
X+ = {AccommodationID, Atype, Price_per_night, Aname, Availability, Docs_Required, DID, AdminID}

Thus, **Primary key = AccommodationID**

The left side of all the FDs in the minimal set of FDs for the relation ‘Accommodation’ is AccommodationID, which is the primary key of this relation, so “Accommodation” is in **BCNF**.

### 17. ‘Activities’ Relation:

Attributes: {ActivityID, Activity_Name, Activity_Type, Price, Duration, Availability, DID, AdminID}

Functional Dependencies:

ActivityID → Activity_Name

ActivityID → Activity_Type

ActivityID → Price

ActivityID → Duration

ActivityID → Availability

ActivityID → DID

ActivityID → AdminID

Let X = ActivityID
X+ = {ActivityID, Activity_Name, Activity_Type, Price, Duration, Availability, DID, AdminID}

Thus, **Primary key =** ActivityID

The left side of all the FDs in the minimal set of FDs for the relation ‘Activities’ is ActivityID, which is the primary key of this relation, so “Activities” is in **BCNF**.

### 18. ‘Refund’ Relation:

Attributes : { Refund_Status, BookingID, TransactionID }

**Primary key =** { Refund_Status, BookingID, TransactionID }

There are no Functional Dependencies in this relation as the only two attributes are Refund_Status,  BookingID, and TransactionID which are the primary key.

Thus, the relation “Refund” is in **BCNF**.