The tables used for our database are as follows :-

```
create table Concession (
    concession_id int auto_increment primary key,
    type varchar(50) not null,
    discount_percent decimal(5,2) not null check (discount_percent between 0 and
100)
);

create table Passenger (
    passenger_id int auto_increment primary key,
    name varchar(100) not null,
    age int not null,
    gender enum('M','F','O') not null,
    concession_id int,
    foreign key (concession_id) references Concession(concession_id)
);

create table station (
    station_id int auto_increment primary key,
    code varchar(10) unique not null,
    name varchar(100) not null,
    how_long_to_wait time not null,
    location varchar(100)
);

create table Route (
    route_id int auto_increment primary key,
    origin_station_id int not null,
    destination_station_id int not null,
    foreign key (origin_station_id) references station(station_id),
    foreign key (destination_station_id) references station(station_id)
);

create table train (
    train_id int auto_increment primary key,
    name varchar(100) not null,
    route_id int not null,
    foreign key (route_id) references Route(route_id)
);

create table trainSchedule (
    train_id int not null,
    station_id int not null,
    stop_number int not null,
    arrival_time time,
    departure_time time,
    primary key (train_id, stop_number),
    foreign key (train_id) references train(train_id),
    foreign key (station_id) references station(station_id)
);

create table Class (
    class_id int auto_increment primary key,
    name varchar(50) not null,
    fare_multiplier decimal(5,2) not null
);

create table trainClassAvailabilityA (
    train_id int not null,
    class_id int not null,
    total_seats int not null,
    primary key (train_id, class_id),
    foreign key (train_id) references train(train_id),
```

```
    foreign key (class_id) references Class(class_id)
);

create table trainClassAvailabilityB (
    train_id int not null,
    class_id int not null,
    travel_date date not null,
    available_seats int not null,
    primary key (train_id, class_id, travel_date),
    foreign key (train_id) references train(train_id),
    foreign key (class_id) references Class(class_id)
);

create table Ticket (
    ticket_id int auto_increment primary key,
    passenger_id int not null,
    train_id int not null,
    class_id int not null,
    travel_date date not null,
    booking_date datetime not null default current_timestamp,
    status enum('BOOKED','CANCELLED','RAC','WAITLIST') not null,
    seat_count int not null,
    base_fare decimal(10,2) not null,
    concession_amount decimal(10,2) default 0,
    net_fare decimal(10,2) as (base_fare - concession_amount) persistent,
    foreign key (passenger_id) references Passenger(passenger_id),
    foreign key (train_id) references train(train_id),
    foreign key (class_id) references Class(class_id)
);

create table Payment (
    payment_id int auto_increment primary key,
    ticket_id int not null,
    amount decimal(10,2) not null,
    payment_mode enum('ONLINE','COUNTER') not null,
    payment_date datetime not null default current_timestamp,
    foreign key (ticket_id) references Ticket(ticket_id)
);

create table Cancellation (
    cancellation_id int auto_increment primary key,
    ticket_id int not null,
    cancel_date datetime not null default current_timestamp,
    refund_amount decimal(10,2) not null,
    processed boolean not null default false,
    foreign key (ticket_id) references Ticket(ticket_id)
);

Here are the various triggers and procedure used for working with the
database :-

Trigger used to update booked status for a given seat -

delimiter $$
create trigger trg_after_ticket_insert
after insert on Ticket
for each row
begin
    if new.status = 'BOOKED' then
        update trainClassAvailabilityB
        set available_seats = available_seats - new.seat_count
        where train_id = new.train_id
          and class_id = new.class_id
          and travel_date = new.travel_date;
```

```
    end if;
end$$
```

Trigger to update availibility after cancellation -

```
create trigger trg_after_ticket_update
after update on Ticket
for each row
begin
    if old.status = 'BOOKED' and new.status = 'CANCELLED' then
        update trainClassAvailabilityB
        set available_seats = available_seats + new.seat_count
        where train_id = new.train_id
          and class_id = new.class_id
          and travel_date = new.travel_date;
        insert into Cancellation(ticket_id, refund_amount)
        values (new.ticket_id, new.concession_amount + (new.base_fare -
new.concession_amount) * 0.5);
    end if;
end$$
delimiter ;
```

Procedure to get PNR status -

```
delimiter $$
create procedure sp_get_pnr_status(in p_ticket_id int)
begin
    select ticket_id, status, travel_date, seat_count, net_fare
    from Ticket
    where ticket_id = p_ticket_id;
end$$
```

Procedure to get train schedule for a given train id -

```
create procedure sp_get_train_schedule(in p_train_id int)
begin
    select ts.stop_number, s.code, s.name, ts.arrival_time, ts.departure_time
    from trainSchedule ts
    join station s on ts.station_id = s.station_id
    where ts.train_id = p_train_id
    order by ts.stop_number;
end$$
```

Procedure to get available seats for a given train, date and class -

```
create procedure sp_get_available_seats(in p_train_id int, in p_class_id int, in
p_travel_date date)
begin
    select available_seats
    from trainClassAvailabilityB
    where train_id = p_train_id
      and class_id = p_class_id
      and travel_date = p_travel_date;
end$$
```

Procedure to get all passengers for a given train on a given date -

```
create procedure sp_list_passengers(in p_train_id int, in p_travel_date date)
begin
    select p.passenger_id, p.name, t.status
    from Passenger p
    join Ticket t on p.passenger_id = t.passenger_id
    where t.train_id = p_train_id
      and t.travel_date = p_travel_date
```

```
      and t.status = 'BOOKED';
end$$

Procedure to get waitlisted passengers for a given train and date -

create procedure sp_get_waitlisted_passengers(in p_train_id int, in
p_travel_date date)
begin
    select p.passenger_id, p.name, t.booking_date
    from Passenger p
    join Ticket t on p.passenger_id = t.passenger_id
    where t.train_id = p_train_id
      and t.travel_date = p_travel_date
      and t.status = 'WAITLIST'
    order by t.booking_date;
end$$
delimiter $$

Procedure to handle cancellation -

create procedure sp_handle_cancellation (in p_ticket_id int)
begin
    declare v_train_id int;
    declare v_class_id int;
    declare v_travel_date date;
    declare v_seat_count int;
    declare v_base_fare decimal(10,2);
    declare v_concession decimal(10,2);

    select train_id, class_id, travel_date, seat_count, base_fare,
concession_amount
    into v_train_id, v_class_id, v_travel_date, v_seat_count, v_base_fare,
v_concession
    from ticket
    where ticket_id = p_ticket_id;

    update trainclassavailabilityb
    set available_seats = available_seats + v_seat_count
    where train_id = v_train_id and class_id = v_class_id and travel_date =
v_travel_date;

    insert into cancellation (ticket_id, refund_amount)
    values (p_ticket_id, v_concession + (v_base_fare - v_concession) * 0.5);

    update ticket
    set status = 'BOOKED'
    where ticket_id = (
        select ticket_id from ticket
        where train_id = v_train_id and class_id = v_class_id and travel_date =
v_travel_date and status = 'RAC'
        order by booking_date
        limit 1
    );

    update ticket
    set status = 'RAC'
    where ticket_id = (
        select ticket_id from ticket
        where train_id = v_train_id and class_id = v_class_id and travel_date =
v_travel_date and status = 'WAITLIST'
        order by booking_date
        limit 1
    );
end$$
```

```
delimiter ;

Procedure to handle refunds -

create procedure sp_total_refund(in p_train_id int, in p_start_date date, in
p_end_date date)
begin
    select sum(c.refund_amount) as total_refund
    from Cancellation c
    join Ticket t on c.ticket_id = t.ticket_id
    where t.train_id = p_train_id
      and date(c.cancel_date) between p_start_date and p_end_date;
end$$

Procedure for calculating revenue between two dates -

create procedure sp_total_revenue(in p_start_date date, in p_end_date date)
begin
    select sum(amount) as total_revenue
    from Payment
    where date(payment_date) between p_start_date and p_end_date;
end$$

Procedure to add to cancellation records -

create procedure sp_cancellation_records()
begin
    select t.ticket_id, c.cancel_date, c.refund_amount, c.processed
    from Cancellation c
    join Ticket t on c.ticket_id = t.ticket_id;
end$$

Procedure to find busiest route -

create procedure sp_busiest_route()
begin
    select r.route_id, count(*) as passenger_count
    from Ticket t
    join train tr on t.train_id = tr.train_id
    join Route r on tr.route_id = r.route_id
    where t.status = 'BOOKED'
    group by r.route_id
    order by passenger_count desc
    limit 1;
end$$

Procedure to produce itemised bill -

create procedure sp_itemized_bill(in p_ticket_id int)
begin
    select t.ticket_id,
           t.base_fare,
           c.fare_multiplier,
           t.base_fare * c.fare_multiplier as class_fare,
           t.concession_amount,
           t.net_fare,
           p.payment_mode,
           p.amount,
           p.payment_date
    from Ticket t
    join Class c on t.class_id = c.class_id
    join Payment p on t.ticket_id = p.ticket_id
    where t.ticket_id = p_ticket_id;
```

```
end$$
delimiter ;

trigger for handling departure_time using time_to_wait-

delimiter $$

create table trainschedule (
  train_id int not null,
  station_id int not null,
  stop_number int not null,
  arrival_time time,
  departure_time time,
  primary key (train_id, stop_number),
  foreign key (train_id) references train(train_id),
  foreign key (station_id) references station(station_id)
)$$

create trigger trg_bs_schedule
before insert on trainschedule
for each row
begin
  declare wait_time time;
  select how_long_to_wait into wait_time
    from station
    where station_id = new.station_id;
  set new.departure_time =
    sec_to_time(
      time_to_sec(new.arrival_time)
      + time_to_sec(wait_time)
    );
end$$

create trigger trg_bu_schedule
before update on trainschedule
for each row
begin
  declare wait_time time;
  select how_long_to_wait into wait_time
    from station
    where station_id = new.station_id;
  set new.departure_time =
    sec_to_time(
      time_to_sec(new.arrival_time)
      + time_to_sec(wait_time)
    );
end$$

delimiter ;

the schema is in 1nf because all the attributes are atomic and indivisible.
the attribute types are:
int
varchar
enum
time
decimal
date
datetime
boolean

the schema has 2 relations with composite keys - trainschedule and
trainclassavailability.
in trainschedule, the key is (train_id, stop_number).
```

the other attributes are station_id, arrival_time, departure_time.
station_id depends on (train_id, stop_number).
arrival_time depends on (train_id, stop_number).
departure_time depends on (train_id, stop_number). ***for abay - this should be refactored by adding an attribute `how_long_to_wait` in station and calculating `departure_time = arrival_time + how_long_to_wait` for each (train_id, station_id)***

in trainclassavailability, the key is (train_id, class_id, travel_date).
the other attributes are total_seats, available_seats.
total_seats depends on (train_id, class_id). it does not depend on travel_date.
available_seats depends on (train_id, class_id, travel_date).

therefore, the schema is not in 2nf.

to convert it to 2nf, we split trainclassavailability into 2 tables - a and b.
a<train_id, class_id, total_seats>
b<train_id, class_id, travel_date, available_seats>

now, the schema is in 1nf.

the schema is in 3nf because there are no transitive dependencies in any of the relations.

the schema is also in 4nf because there are no multi-valued dependencies.

Here are some use cases -

```
+--------------+---------+-----+--------+---------------+
| passenger_id | name    | age | gender | concession_id |
+--------------+---------+-----+--------+---------------+
|            1 | Alice   |  65 | F      |             1 |
|            2 | Bob     |  22 | M      |             2 |
|            3 | Charlie |  30 | O      |          NULL |
|            4 | Diana   |  28 | F      |          NULL |
+--------------+---------+-----+--------+---------------+
4 rows in set (0.001 sec)

MariaDB [hm]> select * from ticket;
+-----------+--------------+----------+----------+-------------
+---------------------+----------+------------+-----------+------------------
+----------+
| ticket_id | passenger_id | train_id | class_id | travel_date | booking_date
| status   | seat_count | base_fare | concession_amount | net_fare |
+-----------+--------------+----------+----------+-------------
+---------------------+----------+------------+-----------+------------------
+----------+
|         1 |            1 |        1 |        1 | 2025-05-01  | 2025-04-24
00:33:34 | BOOKED   |          1 |   1000.00 |            500.00 |   500.00 |
|         2 |            2 |        1 |        1 | 2025-05-01  | 2025-04-24
00:33:34 | RAC      |          1 |   1000.00 |              0.00 |  1000.00 |
|         3 |            3 |        1 |        1 | 2025-05-01  | 2025-04-24
00:33:34 | WAITLIST |          1 |   1000.00 |              0.00 |  1000.00 |
|         4 |            4 |        1 |        2 | 2025-05-01  | 2025-04-24
00:33:34 | BOOKED   |          1 |   1500.00 |              0.00 |  1500.00 |
+-----------+--------------+----------+----------+-------------
+---------------------+----------+------------+-----------+------------------
+----------+
4 rows in set (0.001 sec)

MariaDB [hm]> update ticket
    ->   set status = 'CANCELLED'
    -> where ticket_id = 1;
Query OK, 1 row affected (0.005 sec)
```

```
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [hm]> select * from ticket;
+-----------+-------------+----------+----------+------------
+---------------------+-----------+------------+-----------+------------------
+----------+
| ticket_id | passenger_id | train_id | class_id | travel_date | booking_date
| status    | seat_count | base_fare | concession_amount | net_fare |
+-----------+-------------+----------+----------+------------
+---------------------+-----------+------------+-----------+------------------
+----------+
|         1 |           1 |        1 |        1 | 2025-05-01  | 2025-04-24
00:33:34 | CANCELLED |          1 |   1000.00 |            500.00 |   500.00 |
|         2 |           2 |        1 |        1 | 2025-05-01  | 2025-04-24
00:33:34 | RAC       |          1 |   1000.00 |              0.00 |  1000.00 |
|         3 |           3 |        1 |        1 | 2025-05-01  | 2025-04-24
00:33:34 | WAITLIST  |          1 |   1000.00 |              0.00 |  1000.00 |
|         4 |           4 |        1 |        2 | 2025-05-01  | 2025-04-24
00:33:34 | BOOKED    |          1 |   1500.00 |              0.00 |  1500.00 |
+-----------+-------------+----------+----------+------------
+---------------------+-----------+------------+-----------+------------------
+----------+
4 rows in set (0.001 sec)

MariaDB [hm]> call sp_cancellation_records();
+-----------+---------------------+---------------+-----------+
| ticket_id | cancel_date         | refund_amount | processed |
+-----------+---------------------+---------------+-----------+
|         1 | 2025-04-24 00:35:06 |        750.00 |         0 |
+-----------+---------------------+---------------+-----------+
1 row in set (0.002 sec)

Query OK, 0 rows affected (0.002 sec)

MariaDB [hm]> call sp_get_train_schedule(1);
+-------------+------+-----------+--------------+----------------+
| stop_number | code | name      | arrival_time | departure_time |
+-------------+------+-----------+--------------+----------------+
|           1 | STA  | Station A | 09:00:00     | NULL           |
|           2 | STB  | Station B | 11:00:00     | NULL           |
|           3 | STC  | Station C | 13:30:00     | NULL           |
+-------------+------+-----------+--------------+----------------+
3 rows in set (0.002 sec)

Query OK, 0 rows affected (0.002 sec)

MariaDB [hm]> call sp_itemized_bill(4);
+-----------+-----------+-----------------+------------+------------------
+----------+-------------+---------+---------------------+
| ticket_id | base_fare | fare_multiplier | class_fare | concession_amount |
net_fare | payment_mode | amount  | payment_date        |
+-----------+-----------+-----------------+------------+------------------
+----------+-------------+---------+---------------------+
|         4 |   1500.00 |            1.50 |  2250.0000 |              0.00 |
1500.00 | COUNTER      | 1500.00 | 2025-04-24 00:33:34 |
+-----------+-----------+-----------------+------------+------------------
+----------+-------------+---------+---------------------+
1 row in set (0.002 sec)

Query OK, 0 rows affected (0.002 sec)

 select * from ticket;
+-----------+-------------+----------+----------+------------
+---------------------+----------+------------+-----------+------------------
```

```
+----------+
| ticket_id | passenger_id | train_id | class_id | travel_date | booking_date
| status   | seat_count | base_fare | concession_amount | net_fare |
+-----------+-------------+----------+----------+-------------
+---------------------+----------+------------+-----------+------------------
+----------+
|         1 |            1 |        1 |        1 | 2025-05-01  | 2025-04-24
03:13:49 | BOOKED   |          1 |   1000.00 |              0.00 |  1000.00 |
|         2 |            2 |        1 |        1 | 2025-05-01  | 2025-04-24
03:13:49 | RAC      |          1 |   1000.00 |              0.00 |  1000.00 |
|         3 |            3 |        1 |        1 | 2025-05-01  | 2025-04-24
03:13:49 | WAITLIST |          1 |   1000.00 |              0.00 |  1000.00 |
|         4 |            4 |        1 |        1 | 2025-05-01  | 2025-04-24
03:13:49 | WAITLIST |          1 |   1000.00 |              0.00 |  1000.00 |
+-----------+-------------+----------+----------+-------------
+---------------------+----------+------------+-----------+------------------
+----------+
4 rows in set (0.001 sec)

 update ticket set status = "CANCELLED" where ticket_id = 1;
Query OK, 1 row affected (0.010 sec)
Rows matched: 1  Changed: 1  Warnings: 0

 CALL sp_handle_cancellation(1);
Query OK, 5 rows affected (0.011 sec)

MariaDB [f]> select * from ticket;
+-----------+-------------+----------+----------+-------------
+---------------------+-----------+------------+-----------+------------------
+----------+
| ticket_id | passenger_id | train_id | class_id | travel_date | booking_date
| status    | seat_count | base_fare | concession_amount | net_fare |
+-----------+-------------+----------+----------+-------------
+---------------------+-----------+------------+-----------+------------------
+----------+
|         1 |            1 |        1 |        1 | 2025-05-01  | 2025-04-24
03:13:49 | CANCELLED |          1 |   1000.00 |              0.00 |  1000.00 |
|         2 |            2 |        1 |        1 | 2025-05-01  | 2025-04-24
03:13:49 | BOOKED    |          1 |   1000.00 |              0.00 |  1000.00 |
|         3 |            3 |        1 |        1 | 2025-05-01  | 2025-04-24
03:13:49 | RAC       |          1 |   1000.00 |              0.00 |  1000.00 |
|         4 |            4 |        1 |        1 | 2025-05-01  | 2025-04-24
03:13:49 | WAITLIST  |          1 |   1000.00 |              0.00 |  1000.00 |
+-----------+-------------+----------+----------+-------------
+---------------------+-----------+------------+-----------+------------------
+----------+
4 rows in set (0.001 sec)
```