Assignment: Database Operations for Railway

Management System

Duration: 2 Hours

Scenario:

The Railway Management System is a critical part of the transportation sector that manages trains, schedules, routes, passengers, and ticket bookings. You have been appointed as the database consultant to design and implement a database system that efficiently manages the day-to-day operations of the railway system.

Task 1: Database Creation and Table Setup

1. Create a database named RailwayManagementDB.
Ans: create database railwaymanagmentsystem;

2. Create the following tables to track train information, schedules, routes, passengers, and bookings:

o Trains: To store information about trains (TrainID, TrainName, TrainType, TotalSeats).

o Routes: To store information about routes (RouteID, StartStation, EndStation, Distance).

o Schedules: To store train schedules (ScheduleID, TrainID, RouteID, DepartureTime,

ArrivalTime).

o Passengers: To store passenger information (PassengerID, FirstName, LastName, Age, Email).

o Bookings: To store booking details (BookingID, PassengerID, ScheduleID, BookingDate,

SeatNumber).
Ans:

```
create table Trains(

TrainID int ,

TrainName varchar(50),

TrainType varchar(50),

TotalSeats Int);


Create table Routes(

RouteID int ,
```

```
StartStation varchar(50),
EndStation varchar(50),
Distance int);

Create table Schedules(
ScheduleID int ,
TrainId int,
RouteId int,
DepartureTime DateTime,
ArrivalTime DateTime);

create table Passengers(
PassengerID int ,
FirstName varchar(50),
LastName varchar(50),
Age int,
Email varchar(118));

create table Bookings(
BookingID int ,
PassengerID int,
ScheduleID int,
BookingDate Date,
SeatNumber int);
```

3. Insert sample data into the tables for at least:

o 5 trains

o 3 routes

o 5 schedules

o 10 passengers

o 5 bookings

Ans: insert into Trains(TrainID,TrainName,TrainType,TotalSeats)values

-> (1,'Rajdhani Express','Express',300),

-> (2,'Tejas Express','Superfast',200),

-> (3,'Shatabdi Express','Passenger',250),

-> (4,'Duronto Express','Superfast',150),

-> (5,'Garib Rath','Express',350);

insert into Routes(RouteID,StartStation,EndStation,Distance)values

-> (1,'Delhi','Mumbai',1400),

-> (2,'Kolkata','Chennai',1650),

-> (3,'Jaipur','Ahmedabad',650);

insert into schedules (ScheduleID,TrainID,RouteID,DepartureTime,ArrivalTime) values

-> (1,1,1,'2024-10-20 9:00:00','2024-10-20 21:00:00'),

-> (2,2,2,'2024-10-21 8:30:00','2024-10-21 22:00:00'),

-> (3,3,3,'2024-10-22 6:00:00','2024-10-22 14:00:00'),

-> (4,4,1,'2024-10-23 10:00:00','2024-10-23 22:00:00'),

-> (5,5,2,'2024-10-24 7:00:00','2024-10-24 21:00:00');

Query OK, 5 rows affected (0.01 sec)

insert into Passengers(PassengerID,FirstName,LastName,Age,Email)values

-> (1,'Rajesh','Sharma',45,'rajesh.sharma@specialforce.com'),

-> (2,'Priya','Mehra',32,'priya.mehra@specialforce.com'),

-> (3,'Ankit','Verma',29,'ankit.verma@specialforce.com'),

-> (4,'Kavita','Gupta',40,'kavita.gupta@specialforce.com'),

-> (5,'Arun','Patel',50,'arun.patel@specialforce.com'),

-> (6,'Neha','Joshi',27,'neha.joshi@specialforce.com'),

-> (7,'Suresh','Nair',33,'suresh.nair@specialforce.com'),

-> (8,'Pooja','Reddy',36,'pooja.reddy@specialforce.com'),

-> (9,'Vikram','Singh',42,'vikram.singh@specialforce.com'),

-> (10,'Aarti','Desai',25,'aarti.desai@specialforce.com');

insert into Bookings(BookingID,PassengerID,ScheduleID,BookingDate,SeatNumber) values

-> (1,1,1,'2024-10-10',12),

-> (2,2,1,'2024-10-11',34),

-> (3,3,2,'2024-10-12',56),

-> (4,4,3,'2024-10-13',18),

-> (5,5,4,'2024-10-14',22);

Task 2: Add Constraints After Data Insertion (Strictly write after data insertion)

1. Add a Primary Key to each table.

2. Add Foreign Keys to establish relationships between:

o Schedules and Trains (on TrainID).

o Schedules and Routes (on RouteID).

o Bookings and Passengers (on PassengerID).

o Bookings and Schedules (on ScheduleID).

Ans: ALTER TABLE Trains

ADD CONSTRAINT PK_Trains PRIMARY KEY (TrainID);

ALTER TABLE Routes

ADD CONSTRAINT PK_Routes PRIMARY KEY (RouteID);

ALTER TABLE Schedules

ADD CONSTRAINT PK_Schedules PRIMARY KEY (ScheduleID);

ALTER TABLE Passengers

ADD CONSTRAINT PK_Passengers PRIMARY KEY (PassengerID);

```
ALTER TABLE Bookings
ADD CONSTRAINT PK_Bookings PRIMARY KEY (BookingID);

ALTER TABLE Schedules
ADD CONSTRAINT FK_Schedules_Trains FOREIGN KEY (TrainID)
REFERENCES Trains(TrainID);

ALTER TABLE Schedules
ADD CONSTRAINT FK_Schedules_Routes FOREIGN KEY (RouteID)
REFERENCES Routes(RouteID);

ALTER TABLE Bookings
ADD CONSTRAINT FK_Bookings_Passengers FOREIGN KEY (PassengerID)
REFERENCES Passengers(PassengerID);

ALTER TABLE Bookings
ADD CONSTRAINT FK_Bookings_Schedules FOREIGN KEY (ScheduleID)
REFERENCES Schedules(ScheduleID);
```

Task 3: Joins and Queries

1. Query 1: Write a query to retrieve the train name, route details, and schedule for all trains using an INNER JOIN between the Trains, Routes, and Schedules tables.

Ans:

```
SELECT T.TrainName, R.StartStation, R.EndStation, S.DepartureTime, S.ArrivalTime
-> FROM Trains T
-> INNER JOIN Schedules S ON T.TrainID = S.TrainID
```

-> INNER JOIN Routes R ON S.RouteID = R.RouteID;

TrainName	StartStation	EndStation	DepartureTime	ArrivalTime
Rajdhani Express	Delhi	Mumbai	2024-10-20 09:00:00	2024-10-20 21:00:00
Duronto Express	Delhi	Mumbai	2024-10-23 10:00:00	2024-10-23 22:00:00
Tejas Express	Kolkata	Chennai	2024-10-21 08:30:00	2024-10-21 22:00:00
Garib Rath	Kolkata	Chennai	2024-10-24 07:00:00	2024-10-24 21:00:00
Shatabdi Express	Jaipur	Ahmedabad	2024-10-22 06:00:00	2024-10-22 14:00:00

2. Query 2: Write a query to retrieve all trains that don't have any bookings using a LEFT JOIN between the Trains and Bookings tables.

Ans: select trains.TrainID,trains.TrainName from trains

-> left join schedules on trains.trainid = schedules.trainid

-> left join bookings on bookings.scheduleid=schedules.scheduleid

-> where bookings.scheduleid is null;

TrainID	TrainName
5	Garib Rath

3. Query 3: Write a query to find all passengers who have booked seats for trains traveling a distance of more than 500 km using a RIGHT JOIN and subquery.

Ans: select concat(firstname," ",lastname) as 'Passangers Name',Bookings.PassengerID,Passengers.PassengerID ,Schedules.ScheduleID,Routes.RouteID

-> from Passengers

-> right join Bookings on Bookings.PassengerID=Passengers.PassengerID

-> right join Schedules on Bookings.ScheduleID=Schedules.ScheduleID

-> right join Routes on Schedules.RouteID=Routes.RouteID

-> where Routes.Distance in (

-> select Distance from Routes

-> where Distance>500)

-> order by Passengers.PassengerID;

Passangers Name	PassengerID	PassengerID	distance	ScheduleID	RouteID
NULL	NULL	NULL	1650	5	2
Rajesh Sharma	1	1	1400	1	1
Priya Mehra	2	2	1400	1	1
Ankit Verma	3	3	1650	2	2
Kavita Gupta	4	4	650	3	3
Arun Patel	5	5	1400	4	1

4. Query 4: Write a query to list all train schedules, even if there are no passengers booked, using an OUTER JOIN.

Ans: select s.scheduleid, s.trainid, s.routeid, s.departuretime, s.arrivaltime

-> from schedules s

-> left join bookings b on s.scheduleid = b.scheduleid

-> union

-> select s.scheduleid, s.trainid, s.routeid, s.departuretime, s.arrivaltime

-> from schedules s

-> right join bookings b on s.scheduleid = b.scheduleid;

scheduleid	trainid	routeid	departuretime	arrivaltime
1	1	1	2024-10-20 09:00:00	2024-10-20 21:00:00
2	2	2	2024-10-21 08:30:00	2024-10-21 22:00:00
3	3	3	2024-10-22 06:00:00	2024-10-22 14:00:00
4	4	1	2024-10-23 10:00:00	2024-10-23 22:00:00
5	5	2	2024-10-24 07:00:00	2024-10-24 21:00:00

Task 4: Normalization

1. Normalize the tables to the 3rd Normal Form (3NF) to eliminate redundancy and ensure data integrity.

Ans:
to eliminate redundancy and ensure data integrity we can create one more table for trin types containing train_type_id (primary key) and train type and in trains table instead of train type we can store train type id foreign key referring to primary key (trintypeid) in train type tables. Others are already in normalize form as we have given key to it already

i.e.

TrainID(pk)	Train Name	Train type id(fk)	Total Seats

1	Rajdhani Express	1	300
2	Tejas Express	2	200
3	Shatabdi Express	3	250
4	Duronto Express	4	150
5	Garib Rath	5	350

Train type id(pk)	Train type
1	Express
2	Superfast
3	Passenger
4	Superfast
5	Express

Task 5: Sub Queries

1. Query 5: Write a query to calculate the total number of passengers for each train route.

Ans: select routes.routeid,

-> (select count(distinct bookings.passengerid)from bookings

-> join schedules on bookings.scheduleid=schedules.scheduleid

-> where schedules.routeid=routes.routeID) as 'Total Passengers' from routes;

```
+----------+-------------------+
| RouteID  | TotalPassengers   |
+----------+-------------------+
|        1 |                 3 |
|        2 |                 1 |
|        3 |                 1 |
+----------+-------------------+
```

2. Query 6: Write a query to find the average number of passengers booked per train.

Ans:

select Trains.TrainName, (select avg(totalpassangers) from

(select Schedules.trainid,count(Bookings.passengerid) as totalpassangers

from schedules

join bookings on Schedules.Scheduleid=bookings.Scheduleid

group by Schedules.trainid)as PassengerCountquerry)

as 'Average Passengers Booked' from trains;

TrainName	Average Passengers Booked
Rajdhani Express	1.2500
Tejas Express	1.2500
Shatabdi Express	1.2500
Duronto Express	1.2500
Garib Rath	1.2500

3. Query 7: Write a query to find the train with the highest number of bookings.

Ans: SELECT

-> t.TrainID,

-> t.TrainName,

-> COUNT(b.BookingID) AS TotalBookings

-> FROM Trains t

-> JOIN Schedules s ON t.TrainID = s.TrainID

-> JOIN Bookings b ON s.ScheduleID = b.ScheduleID

-> GROUP BY t.TrainID, t.TrainName

-> ORDER BY TotalBookings DESC

-> LIMIT 1;

TrainID	TrainName	TotalBookings
1	Rajdhani Express	2

4. Query 8: Write a query to find the total seats booked per train route where the booking date is between 01-Jan-2023 and 31-Dec-2023.

Ans:
SELECT

-> r.RouteID,

```
    ->     r.StartStation,
    ->     r.EndStation,
    ->     COUNT(b.SeatNumber) AS TotalSeatsBooked
    -> FROM Routes r
    -> JOIN Schedules s ON r.RouteID = s.RouteID
    -> JOIN Bookings b ON s.ScheduleID = b.ScheduleID
    -> WHERE b.BookingDate BETWEEN '2023-01-01' AND '2023-12-31'
    -> GROUP BY r.RouteID, r.StartStation, r.EndStation;
```

TrainName	Average Passengers Booked
Rajdhani Express	1.2500
Tejas Express	1.2500
Shatabdi Express	1.2500
Duronto Express	1.2500
Garib Rath	1.2500

5. Query 9: Write a query to list all bookings where the passenger's age is greater than 60.

Ans:

```
select b.BookingID, b.PassengerID, b.ScheduleID, b.BookingDate, b.SeatNumber from
bookings b

join passengers p on b.PassengerID=p.PassengerID

where p.PassengerID in (

select p.age from passengers

where p.age>60);
```

Note : as there are no passenger above 60 to check code is correct doing same for age 40

```
select b.BookingID, b.PassengerID, b.ScheduleID, b.BookingDate, b.SeatNumber from
bookings b

join passengers p on b.PassengerID=p.PassengerID
```

where p.PassengerID in (

select p.PassengerID from passengers

where p.age>40);

BookingID	PassengerID	ScheduleID	BookingDate	SeatNumber
1	1	1	2024-10-10	12
5	5	4	2024-10-14	22

Task 6: Stored Procedures and Functions

1. Write a stored procedure to update the number of available seats in a train after a booking has been made.

Ans:

```
alter table Schedules
add column AvailableSeats int;


update Schedules s
join Trains t on s.TrainID = t.TrainID
set s.AvailableSeats = t.TotalSeats;



DELIMITER //
create procedure UpdateAvailableSeatsFromBookings(in input_ScheduleID int)
begin
declare totalSeats int;
declare bookedSeats int;
select t.TotalSeats into totalSeats
from Trains t
join Schedules s on t.TrainID = s.TrainID
where s.ScheduleID = input_ScheduleID;
```

```
select count(*) into bookedSeats
from Bookings
where ScheduleID = input_ScheduleID;
update Schedules
set AvailableSeats = totalSeats - bookedSeats
where ScheduleID = input_ScheduleID;
end //
DELIMITER ;
```

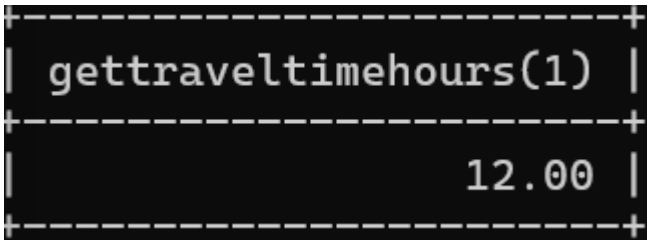
ScheduleID	TrainId	RouteId	DepartureTime	ArrivalTime	AvailableSeats
1	1	1	2024-10-20 09:00:00	2024-10-20 21:00:00	298
2	2	2	2024-10-21 08:30:00	2024-10-21 22:00:00	199
3	3	3	2024-10-22 06:00:00	2024-10-22 14:00:00	249
4	4	1	2024-10-23 10:00:00	2024-10-23 22:00:00	149
5	5	2	2024-10-24 07:00:00	2024-10-24 21:00:00	350

2. Write a function to calculate the total travel time (in hours) between two stations based on departure and arrival times from the Schedules table.

Ans:

```
delimiter //
create function gettraveltimehours(schedule_id int)
returns decimal(5,2)
deterministic
begin
declare traveltime decimal(5,2);
select timestampdiff(minute, departuretime, arrivaltime) / 60.0
into traveltime
from schedules
where scheduleid = schedule_id;
return traveltime;
end //
```

delimiter ;



gettraveltimehours(1)
12.00

Task 7: Views and Indexes

1. Create a view named PassengerBookingsView that combines passenger details, train information, and booking details in one query.

Ans: create view passengerbookingsview as

select

p.passengerid as passenger_id,

p.firstname,

p.lastname,

p.age,

p.email,

b.bookingid,

b.bookingdate,

b.seatnumber,

s.scheduleid,

s.departuretime,

s.arrivaltime,

t.trainid,

t.trainname,

t.traintype,

t.totalseats

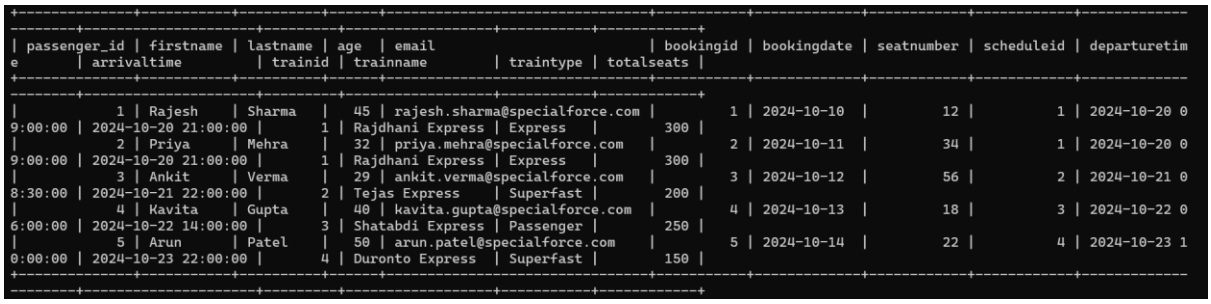
from passengers p

join bookings b on b.passengerid = p.passengerid

join schedules s on s.scheduleid = b.scheduleid

join trains t on t.trainid = s.trainid

order by p.passengerid;



passenger_id	firstname	lastname	age	email	bookingid	bookingdate	seatnumber	scheduleid	departuretime	arrivaltime	trainid	trainname	traintype	totalseats
1	Rajesh	Sharma	45	rajesh.sharma@specialforce.com	1	2024-10-10	12	1	2024-10-20 09:00:00	2024-10-20 21:00:00	1	Rajdhani Express	Express	300
2	Priya	Mehra	32	priya.mehra@specialforce.com	2	2024-10-11	34	1	2024-10-20 09:00:00	2024-10-20 21:00:00	1	Rajdhani Express	Express	300
3	Ankit	Verma	29	ankit.verma@specialforce.com	3	2024-10-12	56	2	2024-10-21 08:30:00	2024-10-21 22:00:00	2	Tejas Express	Superfast	200
4	Kavita	Gupta	40	kavita.gupta@specialforce.com	4	2024-10-13	18	3	2024-10-22 06:00:00	2024-10-22 14:00:00	3	Shatabdi Express	Passenger	250
5	Arun	Patel	50	arun.patel@specialforce.com	5	2024-10-14	22	4	2024-10-23 10:00:00	2024-10-23 22:00:00	4	Duronto Express	Superfast	150

2. Create an index on the Bookings table to improve the performance of queries filtering by BookingDate.

Ans:

create index idx_bookingdate

on bookings (bookingdate);

select * from bookings where bookingdate = '2024-10-10';

BookingID	PassengerID	ScheduleID	BookingDate	SeatNumber
1	1	1	2024-10-10	12

Task 8: Temporary Tables

1. Create a temporary table to store the schedule of all trains departing on a specific day (for example, 15-Oct-2023), and then query it.

Ans:

create temporary table temp_schedule_2024_10_21 as

select *

from schedules

where date(departuretime) = '2024-10-21';

ScheduleID	TrainId	RouteId	DepartureTime	ArrivalTime	AvailableSeats
2	2	2	2024-10-21 08:30:00	2024-10-21 22:00:00	199

Task 9: Cursors

1. Write a basic cursor to iterate over the passengers who have booked more than 5 tickets.

Ans:

delimiter //

```
create procedure show_frequent_passengers()
begin
    declare done int default false;
    declare pass_id int;
    declare pass_cursor cursor for
        select passengerid
        from bookings
        group by passengerid
        having count(*) > 5;
    declare continue handler for not found set done = true;
    open pass_cursor;
    read_loop: loop
        fetch pass_cursor into pass_id;
        if done then
            leave read_loop;
        end if;
        select pass_id as passenger_with_more_than_5_bookings;
    end loop;
    close pass_cursor;
end //
delimiter ;
```

Task 10: ACID Properties and Transactions

1. Ensure the database follows ACID properties by using transactions:

o Begin a transaction before updating seat availability.

o Use a savepoint before deleting any booking records.

o Rollback if an error occurs, ensuring no changes are made to the database.

o Commit the transaction only after all changes have been successfully applied.

Ans:

```
start transaction;
```

```
update trains set totalseats = totalseats - 1
where trainid = 1;
savepoint before_delete;
delete from bookings
where bookingid = 10;
rollback to before_delete;
commit;
```

Task 11: Triggers (Understand which trigger to use when and for what)

1. Create a Trigger that automatically assigns a seat number to passengers when a booking is created.

Ans:

```
delimiter //
create trigger assign_seat_before_insert
before insert on bookings
for each row
begin
    declare next_seat int;
    select ifnull(max(seatnumber), 0) + 1 into next_seat
    from bookings
    where scheduleid = new.scheduleid;
    set new.seatnumber = next_seat;
end //
delimiter ;
```

2. Create a Trigger that updates the total available seats in the train after a booking is confirmed.

Ans:

```
delimiter //
create trigger update_avaliableseats_after_booking
after insert on bookings
```

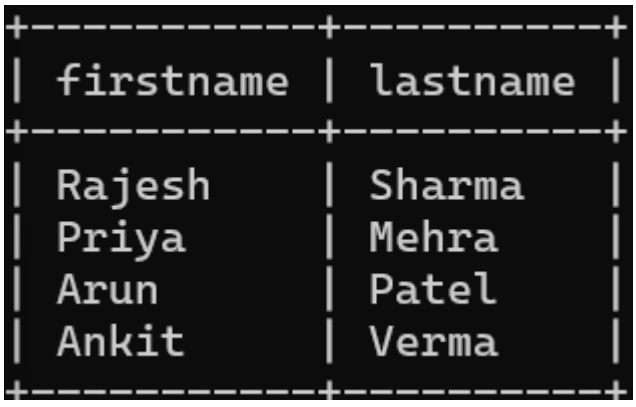
```
for each row
begin
    update trains t
    join schedules s on t.trainid = s.trainid
    set t.availableseats = t.availableseats - 1
    where s.scheduleid = new.scheduleid;
end //
delimiter ;
```

Task 12: UNION and UNION ALL

1. Write a query to combine the results of two queries that return passengers booked on trains for two different routes.

Ans:

```
select p.firstname, p.lastname from passengers p
join bookings b on p.passengerid = b.passengerid
join schedules s on b.scheduleid = s.scheduleid
where s.routeid = 1
union
select p.firstname, p.lastname from passengers p
join bookings b on p.passengerid = b.passengerid
join schedules s on b.scheduleid = s.scheduleid
where s.routeid = 2;
```



firstname	lastname
Rajesh	Sharma
Priya	Mehra
Arun	Patel
Ankit	Verma

2. Write a query to combine the results of all bookings made on different dates.

Ans:

select * from bookings where bookingdate = '2024-10-10'

union all

select * from bookings where bookingdate = '2024-10-11';

BookingID	PassengerID	ScheduleID	BookingDate	SeatNumber
1	1	1	2024-10-10	12
2	2	1	2024-10-11	34

Task 13: Copying Tables

1. Copy the structure of the Passengers table into a new table named OldPassengers.

Ans: create table OldPassengers like Passengers;

Field	Type	Null	Key	Default	Extra
PassengerID	int	NO	PRI	NULL	
FirstName	varchar(50)	YES		NULL	
LastName	varchar(50)	YES		NULL	
Age	int	YES		NULL	
Email	varchar(118)	YES		NULL	

2. Copy all the data from the Bookings table into another table named ArchivedBookings.

Ans:
create table archivedbookings like bookings;

insert into archivedbookings

select * from bookings;

BookingID	PassengerID	ScheduleID	BookingDate	SeatNumber
1	1	1	2024-10-10	12
2	2	1	2024-10-11	34
3	3	2	2024-10-12	56
4	4	3	2024-10-13	18
5	5	4	2024-10-14	22

==========

Additional Assignment: Keys Practice (You should know exact difference and how to use

them)

Duration: 15-20 Minutes

Scenario:

SpecialForce Private Limited has an Employee Management System that stores employee data. The management system uses various keys to maintain data integrity and ensure fast retrieval of information. You are tasked with identifying and implementing different types of keys in the database schema provided below.

Employee Table:

EmpID (PK)	FirstName	LastName	Email	PhoneNumber	Department
101	Rajesh	Sharma	rajesh@specialforce.com	9876543210	HR
102	Priya	Mehra	priya@specialforce.com	8765432109	Finance
103	Ankit	Verma	ankit@specialforce.com	7654321098	IT
104	Kavita	Gupta	kavita@specialforce.com	6543210987	IT
105	Suresh	Nair	suresh@specialforce.com	5432109876	Sales

Assignment Tasks:

1. Super Key:

Identify all possible Super Keys from the Employee table. Remember, a Super Key is any combination of columns that uniquely identifies each record.
Ans: A Super Key is any combination of columns that uniquely identifies a row.

Super Keys in the Employee table:

- EmpID
- Email
- PhoneNumber
- (EmpID, Email),(EmpID, PhoneNumber)

2. Candidate Key:

Determine which of the Super Keys qualify as Candidate Keys. A Candidate Key is a minimal Super Key, meaning it contains no unnecessary columns.
Ans:
A Candidate Key is a minimal super key with no extra attributes.

Candidate Keys:

- EmpID
- Email
- PhoneNumber

3. Primary Key:

What is the Primary Key for the Employee table? Explain why this key was chosen as the Primary Key.
Ans:
Primary Key: EmpID

Why?

- EmpID is system-generated, always unique, never null.
- It's short, stable, and ideal for indexing.
- Best choice to uniquely identify records.

4. Alternate Key:

Identify any Alternate Keys. Alternate Keys are Candidate Keys that were not chosen as the Primary Key.
Ans:

Alternate Keys are Candidate Keys not selected as the Primary Key.

Alternate Keys:

- Email
- PhoneNumber (if it is unique after updating or inserting in further)

5. Composite Key:

Suppose the company wants to track employees working on multiple projects, where both the EmpID and ProjectID are needed to uniquely identify records in a new EmployeeProjects table.

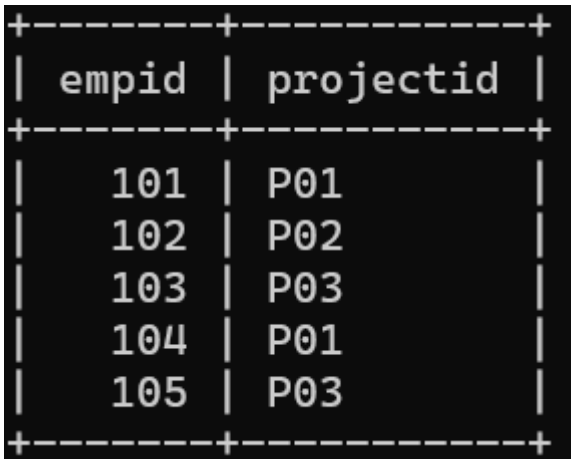
Define a Composite Key for this table using EmpID and ProjectID.

EmployeeProjects Table:

EmpID	ProjectID
101	P01
102	P02
103	P03
104	P01
105	P03

Ans:

```
create table employeeprojects (
    empid int,
    projectid varchar(10),
    primary key (empid, projectid));
insert into employeeprojects (empid, projectid) values
(101, 'P01'),
(102, 'P02'),
(103, 'P03'),
(104, 'P01'),
(105, 'P03');
```



empid	projectid
101	P01
102	P02
103	P03
104	P01
105	P03