Vel Tech
Rangarajan Dr. Sagunthala
R&D Institute of Science and Technology

School of Computing
Department of Computer Science & Engineering
(Artificial Intelligence and Machine Learning)

ACADEMIC YEAR 2025-26 (SUMMER SEMESTER)

LAB RECORD NOTEBOOK

10211CA207 – DATABASE MANAGEMENT SYSTEMS

NAME: G. Bhargav Ram

VTU.NO: 30538

REG.NO: 24UEC40019

BRANCH: CSE (AI-ML)

YEAR/SEM: 2nd Year / IST Sem

SLOT: S10 L12

Vel Tech
Rangarajan Dr. Sagunthala
R&D Institute of Science and Technology

School of Computing
Department of Computer Science & Engineering
(Artificial Intelligence and Machine Learning)

ACADEMIC YEAR 2025-26 (SUMMER SEMESTER)

BONAFIDE CERTIFICATE

NAME : G. Bhargav Ram **BRANCH** : CSECAI-ML

VTU NO. : 30538 **REG.NO.** : 24UECL0019

YEAR/SEM : 2nd year / III sem **SLOT NO.** : S1 0L12

Certified that this is a bonafide record of work done by above student in the **"10211CA207-DATABASE MANAGEMENT SYSTEMS LABORATORY"** during the year 2025-2026 (Summer Semester).

31/10/25

SIGNATURE OF LAB HANDLING FACULTY **SIGNATURE OF HOD**

Submitted for the Semester Practical Examination held on **04.11.25** at Vel Tech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

INDEX

No	Date	Title	Page No.	Marks	Faculty Signature
1.	24/07/25	Conceptual Design after Formal Technical Review	01	18	24/7
2.	31/07/25	Generating design of other traditional database models	04	18	31/7
3.	07/08/25	Developing queries with DML Single-row functions and operations	09	18	7/8
4.	14/08/25	Developing queries with DML Multi-row functions and operations	13	18	14/8
5.	21/08/25	Writing Join Queries, equivalent, and/or recursive queries	18	18	21/8
6.	28/08/25	Writing PL/SQL using Procedures, Function	22	17	28/8
7.	04/09/25	Writing PL/SQL using Loops	25	20	4/9
8.	11/09/25	Normalizing databases using functional dependencies up to BCNF	29	19	11/9
9.	18/09/25	Backing up and recovery in databases	34	17	18/9
10.	25/09/25	CRUD operations in Document databases	37	18	25/9
11.	09/10/25	CRUD operations in Graph databases	42	19	9/10
12.	16/10/25	Micro Project: A gift coupon application that handles offers and payment-related information	47	18	16/10

Total Marks: 218/240

K.T. Thorat

Signature of Faculty

Task-1 Date: 24/07/2025

online temple ticket Booking Management System:

An online temple ticket booking management system allows devotees to book tickets for temple visits, special devotee, poojas and other events online reducing physical queues and improing crowd management. These system often include features like date and time slot selection, payment Processing and the generation of tickets or passes.

Entity:-

The real-world object or Concept that Can be distinctly indentical example Includes students Courses or products.

Entity set:

A Collection of Entities of the same type for instance all students in a university would form an entity set.

Attributes:

A property or characteristic of an entity for example, a student like name, ID and major.

Relationship:

An Association or interaction between two or more entities for example; a students

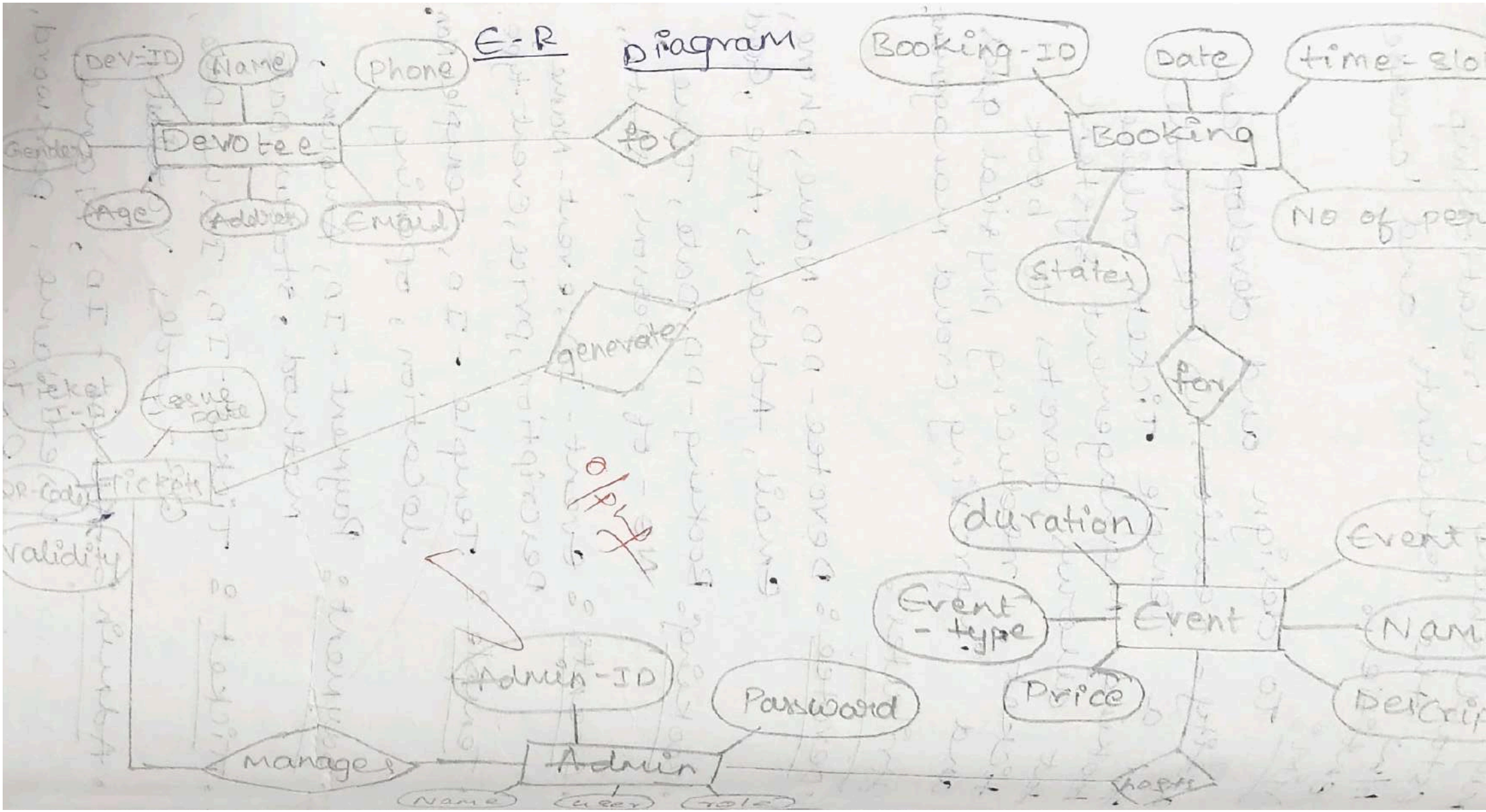
Might enroll in a Course
Establishing a relationship
b/w the 'students' and 'course'
entities.

Aim:

To design and develop an
entity-relationship (ER) Model
for a Temple tickets online
booking Management system
that allows devotes to book
ticket. reducing physical ques
and improving crowd management

Attributes:

- Devotee: Devotee-DD, Name, phone,
email, Addres, Age, Gend
- Booking: Booking-DD, Date, Time,
No-of-person, status,
- Events: Event-ID, event-Name
Description, price, event-type,
- Temple: Temple-ID, Temple-Nam
location, opening
- payment: Payment-ID, Amount,
method, status, Date
- ticket: Ticket-ID, Issue-Date,
QR-Code, Validity.
- Admin: Admin-ID, Name,
UserName, password,
Role.


E-R Diagram
Dev-ID
Name
Phone
Gender
Devotee
Age
Address
Email
to
Booking-ID
Date
time-slo
Booking
No of peo
State
for
generate
Ticket I-D
Issue date
QR-Code
Tickets
Validity
duration
Event
Event - type
Nam
Price
Admin-ID
Password
Manage
Admin
Name
Role

Types of Attributes:

- Simple: Name, price
- Composite: Address (street, City, State)
- Multi-valued: phone
- Derived: Age (from DOB)

Relationship:

- Devotee → Booking (Makes)
- Booking → Event (for)
- Booking → Payment (has)
- Booking → Ticket (generates)
- Event → Temple (hosts)
- Admin → Event (Manages)

Relationship types:

- one-to-one: Booking → Payment, Booking → Ticket
- one-to-Many: Devotee → Boking
- Many-to-one: Booking → Event
- one-to Many: Admin → Event.

Coordinality:

1:N and 1:1 where application

VEL TECH - CSE	
EX NO	5
PE	
RESULT AND ANALYSIS	5
VIVA VOCE (5)	3
RECORD (5)	5
TOTAL (20)	18
SIGN WITH DATE	

Result:

The ER diagram for the temple ticket Booking system was successfully designed showing all entites attributes with correct Cordinalites for database implementation.

(4) 31/7/2025

Task-2 : Generating design of other traditional database Model.

Creating hierarchial / Network Model of the database by enhancing the sound abstract data by Performing following tasks using forms of inheritance.

2a. Identify the specificity of each relationship, find and form suplus relationship.

2b. check is a hierarchy / has a hierarchy and performs generalization and lor specialization relationship.

2c. find the domain of the attribute and Perform a check constraint to the applicable.

2d. Rename the relations.

2e. perform SQL Relations using DDL, DCL Commands.

*. System Context:

A Temple ticket Booking System allows devotees to Book tickets for various temple services (eg. darshan, seras, poojas) either online or at the counter. each booking has details like devotee info, date, type of service, and ticket amount.

(5) Aim:

Creating hioarchioar network Model of the database by enhancing the sound abstract data by performing following tasks using forms of inheritance

2.a. specificity of each relationship & surplus relations.

Entities identified:

- User
- Booking
- Temple
- Pooja
- Payment

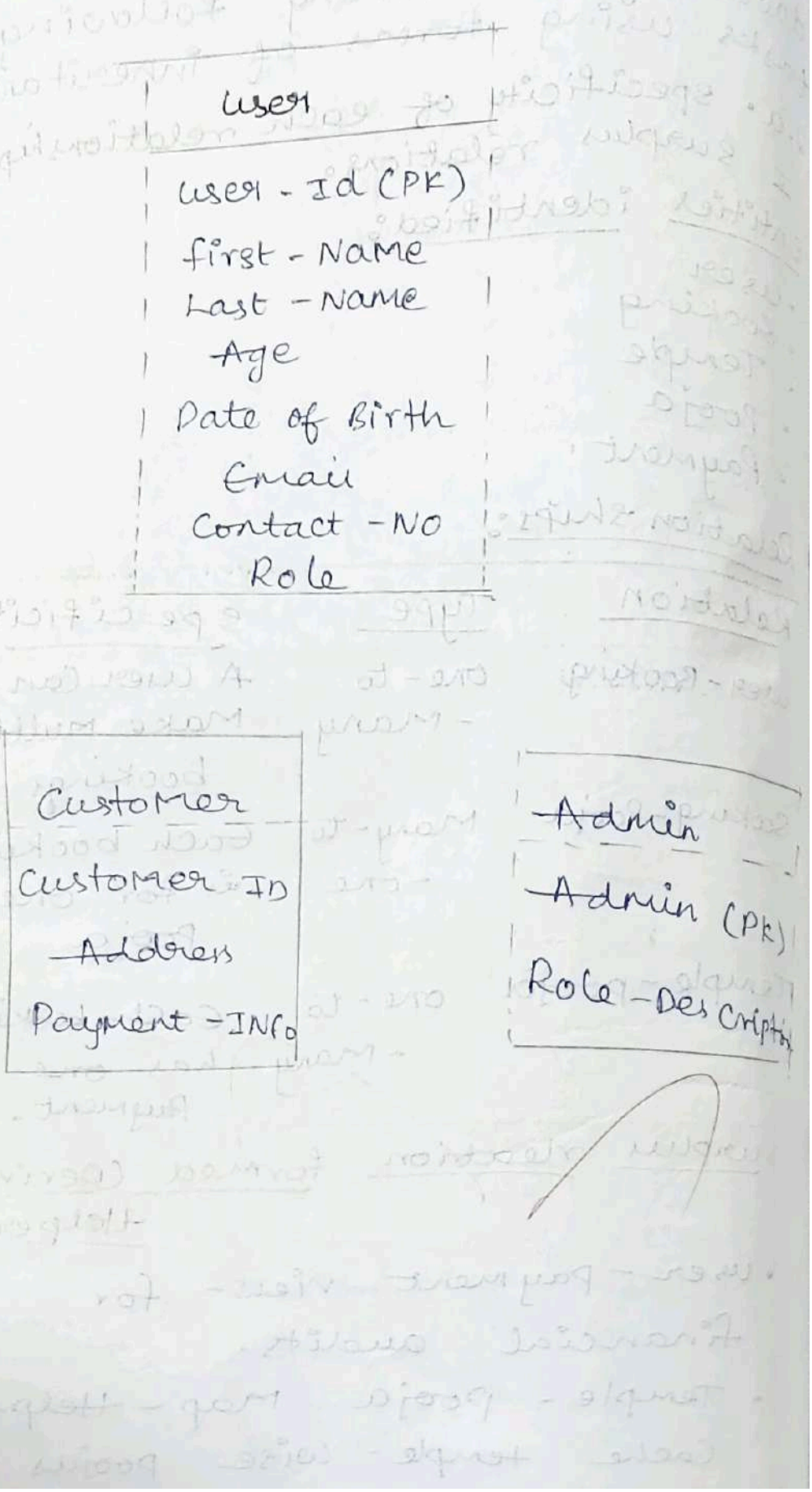
Relationships:

Relation	Type	Specificity
User - Booking	one - to - Many	A user can Make multiple bookings
Booking - Pooja	Many - to - one	Each booking is for one Pooja
Temple - pooja	one - to - Many	Each booking has one Payment.

Surplus relation formed (Derived/ Helper):

- User - Payment view - for financial audits.
- Temple - Pooja Map - Helps Cache temple - wise poojas

1. General Diagram: User as super class for Customer and ~~Admin~~



⑥ 2.b IS-A / HAS-A Hierarchy / Generation & specialization.

IS-A Hierarchy (Generation):

- User
- → admin (IS-A user)
  → Devotee (IS-A user)

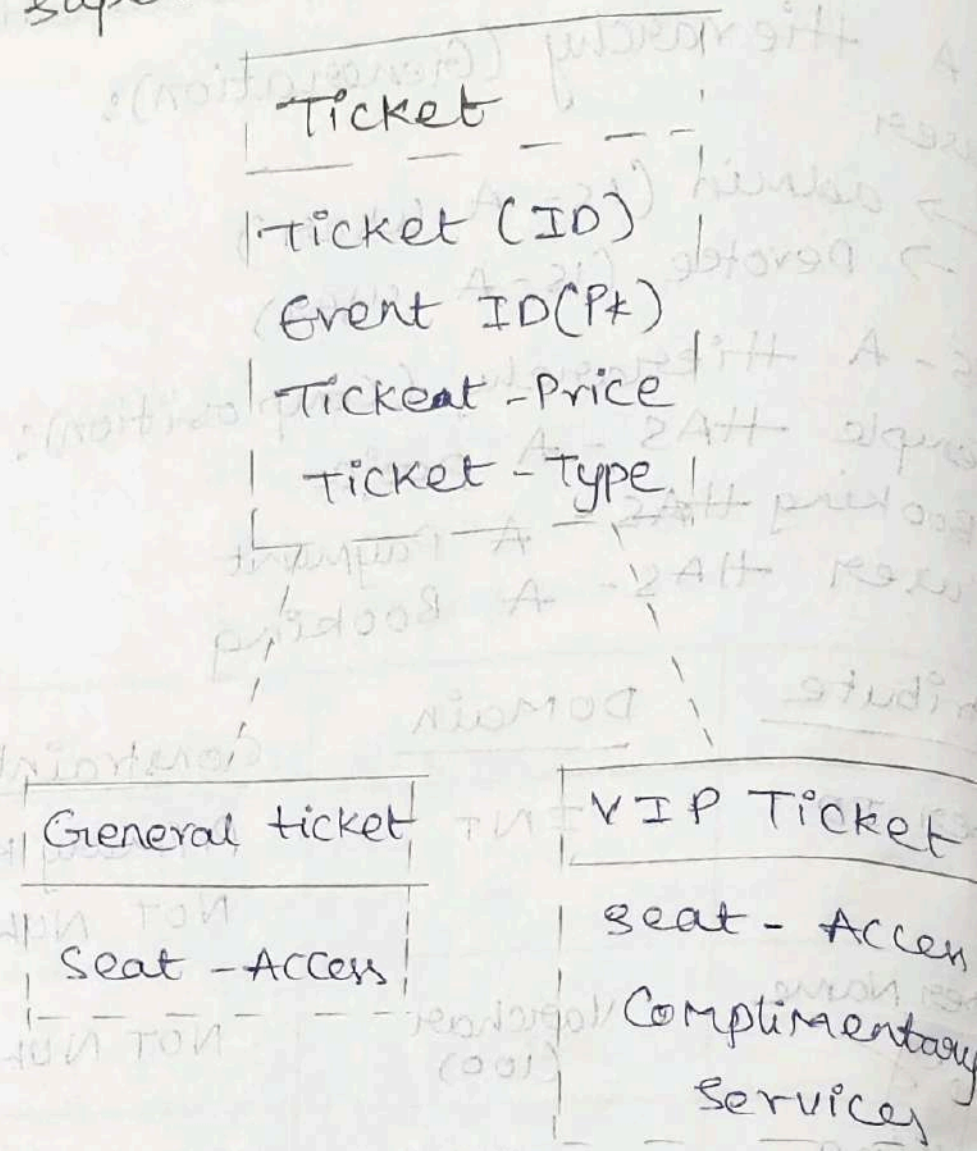
HAS-A Hierarchy (Composition):

- Temple HAS-A pooja
- Booking HAS-A Payment
- user HAS-A Booking

Attribute	Domain	Constraint
User ID	INT	Primary key, NOT NULL
User Name	Varchar (100)	NOT NULL
Booking Date	Date	Check (Booking Date >= Current_date
Amount	Decimal (8,2)	Check (Amount>0)
Poojatime	Time	NOT NULL
Payment Mode	ENUM('UPI', 'credit', Debit', 'Cash')	NOT NULL

2. Specialization

super class



2.d Renaming the relations for clarity:

Old Name	New Name
User	Devotee - Info
Temple	Temple - Master
Booking	Ticket - Booking
Pooja	Pooja - Schedule
Payment	Transcation - record.

2.e SQL Relations using DDL & DCL Commands.

```
Create Table Devotee-Info (User ID
    INT Primary key,
    user name varchar(100),
    user Type ENUM ("Admin',
        Devotee);
Create Table Temple-Master (Temple ID
    INT Primary key)
    Temple Name varchar(150)
    Location varchar(100));
Create Table Pooja-Schedule (
    Pooja ID INT Primary key,
    Temple ID INT,
    Pooja Name varchar(100),
    Pooja Time TIME NOT NULL);
```

⑥ create table Transaction-Record
(Payment ID INT PRIMA
key, Booking ID I
Amount Decimal (8,2)
CHECK (AMOUN
Ticket - Booking (Booking I

DCL- Access control:

- Grant read access to Devotees
Grant select on ticket - Book
- Grant all privilages to Admi
Grant ALL PRIVILEGES ON 'To
Admin - user',
- Revoke DELETE from all
users for safety
REVOKE DELETE ON TICKET-BOOKI

VEL TECH - CSE	
EX NO	9
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	13 ~~15~~ 18
SIGN WITH DATE	

31/04/25

Result:

Thus the Heirarchical Mod
and Network Model for
online ticket Booking
Management System has
been successfully created.

7-8-25

Task-3: using Calcuses, operators and functions in queries

Aim: To perform query processing on a Temple ticket online Booking Management system to different retrival results of queries using DMC, DRC operations with aggregate functions.

Temple.

Temple ID	NAME	LOCATION	CONTACT
T01D01	Tirumalai Venkateshwara	Tirupathi	9024276952
T1D02	Meenakshi Amman	Madurai	7989222275
T1D03	Kashi vish wanath	varanasi	7013982716
T1D04	Jagannath Temple	Puri	9701224404

Visitor:

Visitor ID	f_Name	L_Name	age	Email	Contact_No
V001	Anil	Kumar	30	AnilKumar@gmail.com	9014276954
V002	Akash	Reddy	25	Akash@gmail.com	9701224404
V003	Priya	Sharma	28	Priya@gmail.com	9908290016
V004	Aaron	Patel	35	Aaray@gmail.com	7013982716
V005	Sneha	Rao	22	Sneha@gmail.com	7989222276

10 Booking

Booking ID	Temple ID	Visitor ID	Booking -date	Ticket -Type	amount
B001	T1D01	V001	2024-06-15	VIP	500
B002	T1D02	V002	2024-06-16	General	100
B003	T1D03	V003	2024-06-17	General	100
B004	T1D04	V00H	2024-06-18	VIP	400

Priest:

Priest ID	f Name	L Name	Age	Email	Contact-No
P001	Ramesh	Iyer	50	ramesh@gmail.com	9044276950
P002	Suresh	Sharma	45	suresh@gmail.com	9501229404
P003	Manish	Das	40	Manish@gmail Com	79 89222 785

2. Retrieve details of visitors whose first name starts with 'n'

```
SELECT*
fROM visitor
WHERE fNAME LRKE 'A';
```

Result:

VisitorID	fName	L Name	Age	Contact-n
V002	Akash	Reddy	25	9701224
V00H	Aarav	Patel	35	7013982

3. Add a column for special-seva in Booking table.

```
ALTER TABLE Booking ADD special
Seva VARCHAR(50);
```

Result:

Table altered successfully

4. Count the number of VIP ticket bookings

```
SELECT COUNT(*)
FROM Booking
WHERE Ticket_Type = 'VIP';
```

Result:

Count(*)
2

5. Display temple details for Temple IDS 'TID01', 'TID03', and 'TID04'.

```
SELECT*
FROM TEMPLE
WHERE TEMPLE ID('TID01', 'TI 03, TIDO4');
```

Result:

TempleID	Name	location	Cotact-ID
TID01	Trimula	Tirupati	9701224404
TID02	Kashi	Varanasi	7 01395 2716
TID03	Golden Temple	Amristor	7 98922227

6. Select visitor ID and names of visitor who booked 'special' tickets

```
select visitor ID, fNAME, LNAME
FROM visitor
WHERE visitor ID BNC
```

12

Select VisitorID from Booking where TicketType='Special';

Result:

VisitorID	fName	LName
V005	Sneha	Rao

4. find the priest ID of priest who have not been assigned any temple.

```
SELECT PriestID
FROM Priest
WHERE TempleID is NULL;
```

Result:

PriestID

(NO Result if all priests are assigned)

VEL TECH - CSE	
EX NO	3
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	3
RECORD (5)	5
TOTAL (20)	13+5=18
SIGN WITH DATE	

18/18

Result: Thus, query procesing for temple ticket online Booking management system using clauses operators. was successfully performed.

14-08-25

Task 4:

Using functions in Queries and writing Sub queries:

Aim: To perform advanced query processing and test its heuristic by designing optimal Corrected and tested in the temple ticket online Booking Management System.

4.1: To retrieve all temple details, including the Count of Bookings for each temple.

```
SELECT t.Temple ID,
       t.Name as Temple Name,
       t.location,
       t.Contact.No,
    COUNT(b.Booking ID) As total
                        Booking
fROM TEMPLE t
LEfT   TOIN   Booking b
ON - t.Temple ID = b.Temple ID
Group By t.Temple TD, t.Name,
    t.location, t.Contact_No;
```

Output:

Temple ID	Temple Name	Location	Total Booking
T°1D01	Tirumala	Tirupathi	3
T 1 D02	Menakshi	Madhuri	1
T 1 D03	Kashi	Varanasi	1
T 1 D04	Jagnath temple	Puri	0

4.2 To retrive the Total number of 'special' Seva booking in a temple. wise manner.

```
SELECT t.Name As Temple Name,
COUNT (*) As total special Booking,
fROM TEMPLE t
  TOIN Booking b
  ON t. Temple Rn = b. Temple RD
  WHERE b.T icket-Type = 'special'
Group By t. Name;
```

Output:

Temple Name	Total special Booking
Tirumala venkateshwar	1

4.3: To Retriève the details of temples where bookings include 'VIP' ticket.

```
SELECT*
fROM TEMPLE
WHERE TEMPLE D DNC
        SELECT TEMPLE DD
        fROM Booking
        WHERE Ticket-TYPE='VIP'
);
```

(15)

Output:

Temple ID	Name	Location	Contact-No
TID01	Tirumala	Tirupati	9701224467
TID02	Kasi	Varanasi	7013982716

H.4: To retriere visitors and booking details of visitors who are above 25 years old.

```
SELECT V.visitor ID
       V.f Name AS visitor Name,
       V.age,
       b.Booking ID,
       b.Ticket-Type
       b.amount
FROM visitor V
JOIN Booking b
ON V.visitor DD = b.visitor RD
WHERE V.Age = 25;
```

Output:

Visitor ID	Visitor Name	Age	Booking ID	Amount
V001	Anil	30	B001	500
V003	Priya	28	B003	100
V004	Aaran	35	B004	700

H.5: To retriere the details of temples with no bookings

```
SELECT *
FROM TEMPLE
WHERE TEMPLE ID NOT IN(
 SELECT  Temple ID
 FROM Booking
 );
```

Output:

TempleID	Name	Location	Contact-No
TID04	Jagannath Temple	Puri	9701224404
TID05	Golden Temple	Amritsar	9044276954

H.6: To retriere the templeid, Temple name, and visitor name for a particular visitorial given

```
SELECT t. Temple ID
       t. Name As Temple Name,
       V.f Name As visitor
                   Name
FROM Temple.t
JOIN Booking b
ON t. Temple ID = b.Temple.ID
 JOIN visitor v
ON b. visitor ID = K.Visitor ID
WHERE  V. visitor ID = 'V005';
```

Output:

TempleID	TempleName	visitor Name
T1001	Tirumalar	Snchar

VEL TECH - CSE	
EX NO	4
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	3
RECORD (5)	5
TOTAL (20)	13+5=18
SIGN WITH DATE	

14/8

Result: Thus, the queries using function, pons, and nested subquires were succesfully executed for the temple ticket online Booking management system.

Temple	Pooja Name	Price
Madhuri temple	special Darshnam	250
Tirupathi temple	suprabhata seva	300
Tirupati temple	Archana	100
kanchi temple	Abhisekham	500

Output:

BookingID	Devotee	Pooja Name	Booking date
B101	Rajesh	suprabhata seva	22-6-23
B102	Meena	Archana	22-6-23
B103	Arjun	Abhishekham	23-6-23
B104	kavitha	special dharshnam	23-6-23

21/8/25

Task-5: writing Join Queries, equivalent and or recursive Queries.

(Tools: SQL1 Oracle, ALM: flapped classroom)

Aim:

To perform advanced Query Processing and test its heuristics in Join Queries equivalent Queries, and recursive Queries for temple Ticket online Booking management System.

Queries & outputs:

5.1 To retrieve all temples and their poojas.

Sql:

```
SELECT t.Name As Temple, P=Pooja Name,
P.Price
fROM Temple t
JOIN Pooja P ON t.Temple ID
   = P.Temple ID.
```

5.2 To list all booking along with the devotee and pooja details.

Sql:

```
SELECT b.Booking ID, d.fName As
Devotee, P.PoojaName, b.bookingDate,
b.slottine, b.status
fROM Booking.b
JOIN Devotee d ON b.Devotee ID
   = d.Devotee ID
```

Output:

Devotee	Total Bookings
Rajesh	1
Meena	1
Arjun	1
Kavitha	1

Output:

DevoteeID	fName	Mobile No
D103	Arjun	9876543210
D105	Suresh	8765432190

Pooja Name	Total bookings
Suprabhata Seva	1
Archana	1
Abhisekam	2
Special darshan	1

JOIN Pooja P ON b. Pooja ID
= P. Pooja ID.

5.3 Count the number of bookings made by each devotee.

Sql:

```
SELECT d.fName As Devotee, Count
(b. Booking ID) As Total Booking
fROM Devotee d
LeFT JOIN Booking b ON d.devoteeID
=b. Devotee ID
GROUP BY d. f Name;
```

5.4 To find all devotees who booked 'Abhishekam'

Sql:

```
SELECT d. Devotee ID, d.fName,
d. Mobile No
fROM Devotee.d
JOIN BOOKING b ON d. Devotee
ID = devotee ID
JOIN Pooja P ON b. Pooja ID =
P. Pooja ID
WHERE P. Pooja Name = 'Abhishekam';
```

5.5 To retrieve all poojas and the number of frames they were blocked.

Sql:

```
SELECT P. Pooja Name, Count (b. Booking ID)
As Total Bookings;)
from Pooja P
LETT JOIN Booking b ON P. Pooja Id
= b. Pooja ID
```

Group By P.Pooja Name;

5.6: To retriere the total number of Cancelled booking; temple wise

Sql:

```
SELECT t.Name As Temple, count
(b.Booking ID) As Cancelled Booking;
from TEMPLE.t
JOIN Booking b ON t.temple ID
                = b.Temple ID
WHERE b.Status = "Cancelled"
GROUP BY t.Name;
```

5.7 To retriene temple details where booking were succesfull

Sql:

```
SELECT Distinct t.temple ID, t.Name;
fROM TEMPLE.t
JOIN Booking b ON t.templeID = b.templeID
WHERE b.status = 'conformed'.
```

5.8 To retrieve devotee and booking details for devotees above 50 years old

Sql:

```
SELECT d.fName, d.Age, b.BookingID,
b, 'booking Date, b.Slot time;
fROM Devotee.d
    JOIN Booking b ON d.devotee.ID:
    b.Devotee ID
WHERE d.Age > 50.
```

5.9 To retriene the details of temples where no pooja Booking are done.

output:

TempleID	Name	Location
T00H	Rameshwaram Temple	Ramesh Coaram

output:

Temple	Pooja Name	Devotee	Booking Date	S
Tirupathi Temple	Archana	Meena	22-JUN-2023	9

Sql:
SELECT *
FROM Temple
WHERE Temple ID NOT IN (SELET
Temple ID FROM BOOKING);

5.10: To retrieve temple, Pooja and devotee details for a grong Booking ID.

Sql:
SELECT t.Name As Temple, P. Pooja, Name, b. booking Date, b. slot time
FROM Booking b
JOIN Temple.t ON b. Temple ID
= t. Temple ID
JOIN Devotee d ON b. devotee ID = d. devotee ID
WHERE b. Booking ID = 'B102';

VEL TECH - CSE	
EX NO	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	3
RECORD (5)	13+5=18
TOTAL (20)	
SIGN WITH DATE	

21/8/25

Result: Thus, the temple online Booking Management System queries using Join Queries, equeries were excuted sucefully to Manage temples Devotees, priests and booking.

28-8-25

Task 6: Procedures, Function, and loops using PL/SQL on Number Theory and Business scenarios.

Aim: To write and execute PL/SQL procedures functions and loops or Number Theory and Business scenarios.

Algorithms:

A. Number Thory example- prime number check using procedure

1. Start
2. Read a Number n.
3. Initialize Counter
4. use loop for check divisibility
5. If divisible → Not prime, else
6. Display result

B. Business scenario Example salary Bonus Calculation using function.

1. Start
2. Create a function that accepts emp - salary
3. If salary < 2000 → 20% bonus
   if salary b/w 2000 - 50000 - 10% bonus
   Else → 5% bonus.
4. Return final salary with bonus
5. End.

Program:

A. Number Theory - Prime Number Check SET

```
CREATE OR REPLECE PROCEDURE
        CHECK_ PRIME (n IN NUMBER);
flag BOOLEAN := TURE;
BEGIN
 If n <= 1 THEN
    DBMS- OUTPUT. PUT- LINE (n ll 'is NOT
                                Prime');
Else
  FOR i IN 2. n/2 loop
     If MOD (n,i) = 0 THEN
        flag := FALSE;
         EXIT;
     END LOOP;
     If flag THEN
      DBMS - output.put - LINE
                    (n ll 'is prime');
     Else
      DBMS- OUTPUT. PUT- LINE(n ll 'is
                    NOT prime);
     END If;
    END If;
   END;
  / Execution
BEGIN
    CHECK - Prime (29);
    CHECK - prime (20);
     END;
     /
```

output:

FOR Prime Number Check:

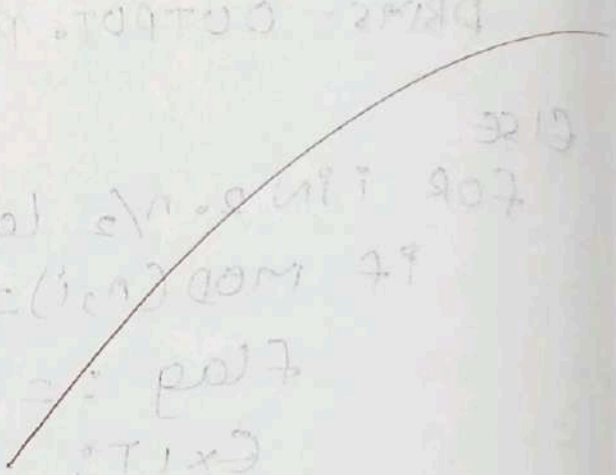
29 is prime

20 is NOT Prime

for Bonus Calculation:

final salary with Bonus: 21600

final salary with Bonus: 33000



24 8. Business scenario - Employee bonus Calculation.

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE FUNCTION
                CALCbonus (salary Number)
RETURNS NUMBER is
   bonus Number;
BEGIN
   If salary < 2000 THEN
         bonus := salary 0.20;
Elsif salary Between 20000 AND
                  50000 THEN
      bonus := salary * 0.10;
Else
      bonus = salary * 0.5;
      END If;
      RETURN (salary + bonus);
END;
-- execution.
DECLARE
   final_sal Number
BEGIN
final_sal := Calc_bonus (3000);
DBMS_output.PUT_LINE (final
                Salary with Bons:
END;
}
```

EX NO	6
ERFORMANCE (5)	4
RESULT AND ANALYSIS (5)	4
VIVA VOCE (5)	4
RECORD (5)	4
TOTAL (20)	20
SIGN WITH DATE	04.09.24

Result:

Thus PL/SQL procedures functions and loops were successfully implemented for Number Theory and Business scenarios.

4-9-25

Task 7: Triggers, Views and Exceptions

Aim: To conduct Events views and exception handling on CRD Operations for the Temple ticket online Booking managem

a) Trigger:

Requirement: when a newbooking is inserted a into the Booking Table, automatically inser a 'Activi'

Sql:

```
CREATE OR REPLACE TRIGGER
trg_insert_ticket
After INSERT ON Booking
        for EACH ROW
BEGIN
    INSERT INTS TICKET
(TicketID, BookingID, QRCode, Status)
    values(
Seq_ticket_irertval
NTH BOOKING ID;
    IDN KNOW;
        NULL;
        NULL,
    SYS_ YUID()
        'Active'
        );
    END;
    )
```

26

D) View

Requirement: Create a view for display Booking detail along with temple & slot time.

```
CREATE OR REPLACE VIEW Booking
Detail View As
    SELECT
    b.BookingID,
    b.Booking RET,
    v.Name As Devotee
    df Name As Darshan name.
    df Name As Darshan type,
    s.statis As slot start,
    s.End TS As slot End,
    b.ots,
    b.Amount,
    b.statey.
    FROM booking b
    Join temple t ONb.temple ID
                = t.temple ID
    Darshan type df on s.Darshan
                Type ID.
    df.Darshan Type ID;
    Select * from Booking Details in
    View;
```

c) Non - Recursive PL/SQL produce

Requirement: Retriere even Number a. Booking IDs any given temple

```
CREATE OR REDIACE PROCEDORE
 get Even Booking IDS
      in_ temple_ id Num
      Sys_ ID C I NumberIST
    )As
    BEGIN
out_ even . booking_ ids: =
      sys. DCI NUMBER LIST();
      for recINC
        SELECT Booking ID
        from Booking
    WHER Temple ID= in_temple.id
        And MOD(Booking ID):0
    End loop;
    End;
    )
```

Execution BLOCK:

```
DECLARE
  temple_ id NUMBER := -101;
  even . booking _ ids sys. OPCINUM
                    _ BER List;
  BEGIN
    get Even Booking ids Per
          temple (
    temple_id even . booking _ ids);
```

```
for CIN, --- even_id.COUNT loop
    DBMS.output((even)
    Booking ID:) ||even booking||)
BEGIN
    Yet even Booking ID's for
        Temple (temple_id,
                even_booking_id
    ||even_booking_ids(i))
        END loop);
        END;
    )
```

VEL TECH - CSE	
EX NO	7
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	5+5+5+5=20
SIGN WITH DATE	04.09.24

Result: Thus, the triggers view and exceptions for the temple ticket online Booking management systems were successfully implemented and verified.

Date: 11-9-25

Task 8: CRUD operations in Document Database

Aim: To perform mongoose using NPM design on mongoDB designing document database and Performing CRUD operations.

Algorithm:

Step 1: Install mango db using following link.
https://www.mongodb.com/try/download/community.

Step 2: Intall mangosh using the below link and install - mangosh.

Step 3: To add the mongoDB shell binary's location to your PATH environment variable.

3.1 open the Control panel

3.2: In the system and security Category, click system.

3.3: Click Advanced system settings.

3.4: Click Environment Variables.

3.5: In the system Variables sectio Select path and click Edit.

3.6: Click new and add the filepath to your mongosh binary.

3.7: Click OK to confirm your changes. your changes

8.1 output:

```
{ "acknowledged": true,
"Inserted Ids": [
ObjectId("651d1cc36ebbfe7993adf911",
ObjectId("651d1cc36ebbfe7993adf912"
]
}
```

8.2 output:

```
{
"acknowledged": true,
"inserted Id": ObjectID("651d1
ce06ebbfe7993adf913")
}
```

step4: To confirm that your PATH environment variable is correctly configured to find mongosh.

step5: open mongo shell 4.0 from C:\Program files\MongoDB\server\bin\Mongod.exe.

step6: Type the CRUD(CREATE READ UPDATE DELETE) ~~CR~~ TEXT FILE. CROD OPERATIONS:

db.create collection("circterBo rad")

- Procedure:

8.1. Insert temples:

```
db. Temples. ~~Enter~~ Inter one({
    temple ID: "T1D01",
    Name : "Meenakshi AmmaTemple",
    Location: "Madurai",
    Contact: 987 654 3210
    Timings: "6:00AM - 9:00PM"
});
```

8.2. Insert ticket Booking

```
db. Tickets. Insert one({
    Ticket ID: "TK1D02",
    Temple ID: "T1D01",
    User name : "John doe"
    Booking Date: "2025-09-10",
    Number of Persons: 3
});
```

3) 5.3.3 Query All Temples (output):


```
{
  "_id" : ObjectId("651d1c986ebbfe
                    7993adf910"),
  "Temple ID" : "TID01",
  "Name" : "Manakshi Temple",
  "Location" : "Madhurai",
  "Contact no" : 9876543210,
  "Timings" : "6:00AM to 9:00PM"
}
{
  "_id" : ObjectId("651d1cc36ebb
                    fe7993adf911"),
  "Temple ID" : "TID02",
  "Name" : "Ramanathaswamy Temple",
  "Location" : "Rameswaram",
  "Contact" : 9765432109,
  "Timings" : "5:00AM - 8:00PM"
}
{
  "_id" : ObjectID("651dcc36ebb
                    fe7993adf912"),
  "Temple ID" : "TID03",
  "Name" : "Brihadeeswarar
            Temple",
  "Location" : "Thanjavur",
  "Contact" : 9654321098,
  "Timings" : "7:00AM - 8:30PM
}
```

8.4. query Ticket Booking by User Name

```
db.tickets.find({ userName : "John Doe"})
```

8.5: Update Temple Contact:

```
db.Temples.updateone(
    { TempleID: "TID02" },
    { $set: { Contact: 9998887776 } }
);
```

8.6: Delete a Ticket Booking

```
db.Tickets.deleteone({
    TicketID: "TKID01" })
);
```

8.7: Delete a Temple

```
db.Temples.deleteOne({
    TempleID: "TID03" }
);
```

8.4: output

{"_id": object id("651d1ee06ebb5e7499
39df");
"ticket-id": "TICD01";
"temple-id": "TID01";
"user name": "John Doe";
"Booking Date": 2025-2-4;
"Number of Periods": 3}
);

8.5: ~~update~~ output: ~~temple~~ ~~Contact~~:

{"acknowledged": True,
"matered count": 1;
"modified out": 1.
}

8.6: output:

{"acknowledged": True,
"deleted arerf": ;
}

8.7: output:

{"acknowledged": True,
"deleted count": 1}

VEL TECH - CSE	
EX NO	8
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	4
RECORD (5)	5
TOTAL (20)	14+5=19
SIGN WITH DATE	D.Mithr 11.09.24

Result: Thus, using MongoDB with mongoose, we successfully designed a document DB for an online temple ticket Booking management system using CRUD operation was successfully executed

Date: 18/09/25

Task-9: Crud operations in Graph Database

Aim: To perform CRUD operations like creating, inserting, querying, updating, and on graph online Temple Ticket Booking.

Algorithm:

Step1: open https://neo4j.com/cloud/platform/aura-graph-database/?ref=docs-gets-started-dropdo

Step2: Click start free → Continue with google → open

Step3: Copy the downloaded Password file info and login to the Aura Console

Step4: start executing queries in the Neo4j Browser.

Output:

d.Devotee_ID	d_Name	dAge	Phone
D001	Raj Kumar	30	9876

Output:

t.templeID	t.Name	t.Location
T001	Sri venkateshwar temple	Tirpati

Output

b.BookingID	b.DevoleeID	b.templeID	b.Date
B001	D001	T001	2021-9-10

- Create a Devotee Node

```
CREATE (d: Devotee {DevoteeID:
'D001', Name: 'Raj Kumar',
Age: 30, Phone: '9876543210',
Email: 'rajkumar@gmail.com'})
RETURN d
```

- Create a Temple Node

```
CREATE (t: Temple {TempleID:
'T001', Name: 'Srivenkateswara Temple',
Location: 'Tirupati',
Capacity: 5000}) RETURN t
```

- Create a Booking Node

```
CREATE (b: Booking {BookingID:
'B001', DevoteeID: 'D001', TempleID:
'T00', Date: '2025-09-10',
No of Tickets: 2}) RETURN b
```

2. Create Relationships

- Devotee makes Booking

```
MATCH (d: Devotee {DevoteeID:
'D00'}), (b: Booking {BookingID: 'B001'})
CREATE (a) - [: MADE] -> (b)
RETURN d, b
```

output:

b.BookingID	b.DevoteeID	b.TempleID	b.date
B001	D001	T001	2024-9-1

output:

d.DevoteeID	d.Name	d.Age	d.phone
D001	Rajkumar	31	9712345641

- Booking linked to Temple

```
MATCH (b:Booking {BookingID:
'Boo1'}), (t:Temple {TempleID:
'To01'})
CREATE (b)-[:FOR]->(t)
RETURN b,t
```

3. Display All Nodes

```
MATCH(n) RETURN n
```

4. Retrieve Particular Booking

```
MATCH (b:Booking {BookingID:
'Boo1'}) RETURN b
```

5. Delete Particular Booking

```
MATCH (d:Devotee {DevoteeID:
'D001'}) SET d.Phone = '9123456780',
d.Age = 31 RETORN d
```

6. Delete Particular Booking

```
MATCH (b:Booking {BookingID:
'Boo1'}) DELETE b
```

VEL TEC...	
EX NO	9
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	2
RECORD (5)	5
TOTAL (20)	17
SIGN WITH DATE	

18/9/25

Result: By following the above steps, was successfully created, inserted, queried, updated and deleted of Temple Ticket Booking Management.

Date: 25/09/25

Task 10: Normality statics using functional dependence upto third normal form.

Aim: To analize the below relation and create the simplified tables with suitable using Temple ticket booking Management system.

* Given Relation:

Temple booking {

Booking ID, User ID, UName, UEmail, U Contact, Temple ID, Temple Name, T locations, Priest-ID, priest Contact, Ticket ID, Booking date Clash data, Time slot, Ticket type, Price, Payment ID, Payment mode, Payment status.

*. step (a): apply functional Dependence -> IName size to 1NF first, identify functional dependencies (FDS):

User ID → U Name, U Email, U Contact
Temple ID → T Name, T contact
Priest ID → Priest Name, priest Contact
Payment ID → payment mode, Payment Status, Transiston Data

* Step (b): Normalize using fDt and 2 t Candidate key:
Booking ID (Main unique Identify of a booking)
from dt (closer of Candidate key):

- Booking ID * {All attributes}
  → Confirms Booking ID is Candidate key
- User ID, Temple ID, Priest ID, Ticket ID, Payment ID also act as foreign key with their own dependencies.

Step (c): Minimal Cover (Conical we reduce fDs to minimal form:

1. User ID → U Name, U Email, U cont
2. Temple ID → T Name, T location
3. Priest ID → Priest Name, Priest Contact
4. Ticket ID → Ticket Type, price

*. Temple Table

Temple (Temple ID [PK], TName, TLocation)

*. priest Table

Priest (Priest ID [PK], Priest Name, Priest Contact.

*. Ticket table

Ticket (Ticket ID [PK], ticket type, Pric)

*. Booking Table

Booking (Booking ID [PK], user ID [FK], Temple ID [FK], Priest ID [FK], Ticket ID [FK], Payment ID [FK],

Constraints:

- Primary key: Booking, user ID, Temple ID, priest ID, Ticket ID, Payment ID.
- foreign key:
  - Booking user ID -> user.user ID
  - Booking Temple ID -> Temple. Temple ID
  - Booking. Priest ID -> Priest. Priest ID
  - Booking. Ticket ID -> Ticket. Ticket ID
  - Booking. Payment ID -> Payment. Payment ID

Unique Constraints: UEmail, UContact

Check Constraints:

- Price >0
- Payment Status {"Pandy", "Completed", 'failed'}
- Timeslot within Valid darshan timings.

VEL TECH - CSE	
EX NO	10
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	3
RECORD (5)	5
TOTAL (20)	18
SIGN WITH DATE	

25/9/25

Result:

Thus, the given relation for the online Temple ticket Booking management System has been normalized into simplified tables up to Third Normal form (3NF).

9/10/25

Task 11:

Aim: to design on online Temple ticket booking management system using oracle forms, menus.

Design the data model:

In oracle forms, define the data model that connects to the database schema for the temple ticket Booking system.

- Temples
- Ticket types
- Booking Details
- users/Devotees
- Payment Information.

Create menu:

menus provide the navigation structure for the Booking system application.

Steps to create menu in oracle forms:

1. open oracle forms Builder.
2. Create a new menu form or use an existing one.

3. Add menu items for each major function.
   - Book Tickets
   - View Booking History
   - Cancel Tickets.
   - manage Temples
   - Reports

4. Define menu hierachy
5. Assign triggers or procedures to handle menu item action.

Design forms:

forms are used to capture, display and edit data related to ticket booking.

steps to design forms in oracle forms:

1. Create a new form for each major component of the booking system.
   - Ticket booking form
   - User Registration form.
2. Add form elements like text fields, buttons and lists.
3. use the property palette to configue element properties.

5. Test the forms inside oracle forms builder for functionally and data accuracy.

CREATE Reports:

Reports provides summarized information above bookings and temple visits.

Steps to create reports using order Report builder:

1. open oracle Report Builder.
2. Create a new reporter or use an existing tempate
3. Design the report layout with headers, footers and detailed data columns.
4. Add parameters to allow filtering.
5. Generate and preview the report.

VEL TECH - CSE	
EX NO	11
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	4
RECORD (5)	5
TOTAL (20)	19
SIGN WITH DATE	

9/10/25

Result: Thus, the designing on online Temple Ticket booking management system with oracle forms, menus, and Report Builder.

Date: 16/10/25

4T

Use Case 3: A gift coupon application that handles offers and payment-related information.

Choose the database system, a relational database, for its capability to scale horizontally while keeping the ACID property. The last was particularly important because transactional data was being handled. Eventual consistency as offered by other databases was not suitable. Going with DB and cluster gives the opportunity to scale horizontally for a large number of writes and reads without compromising the ACID and transactional capability required by the application. Will it transact and lead to no deadlock, keeping all relational tables normalized? If so what normal form do they sustain to offer a gift coupon application?

Team Members:

NEELISETTY VENKATA SAI THARUN

SUNKESWARA SUSHANTH

SHAIK GOUSE MOHIDDIN

DODLA SANTHOSH

MANNURU MEDHINI

KALLURI PEDDA BABU

CHANDRAJIT KAMALAKKANNAN

VADAMADHURA DILEEP KUMAR

YARASANI TARUN KUMAR REDDY

KOMMINA SRUTHI

GOSU BHARGAV RAM

SHAIK SHAHIL ROHAN

THONDAPU SAI JASWANTH

DASARI KOTISURYA

RAVURI NAGA DURGA PRASAD

BOARD OF SECONDARY EDUCATION ANDHRA PRADESH

DADIPINENI BHARGAVI

PANAGANTI SAI SRINATH

AIM:

Build a backend component that:

1. Stores coupons/offers and payment records.
2. Validates & applies coupons to orders (checks expiry, usage limits, min-order, allowed users).
3. Calculates final payable amount and records the payment + coupon usage.

ALGORITHM (high level)

1. Receive order request: (user_id, cart_amount, coupon_code, payment_method).
2. Load coupon by code. If not found → reject.
3. Validate coupon:
   - check active flag
   - check current_date ≤ expiry_date
   - check total_usage and per_user_usage limits
   - check min_order_amount
   - check if user is allowed (optional)
4. Compute discount:
   - if type = percentage → discount = min(cart_amount * pct/100, max_discount)
   - if type = fixed → discount = fixed_amount
5. Final_amount = max(cart_amount - discount, 0) + taxes/shipping (if any).
6. Process payment via gateway (simulate). If success:
   - create payment record (status = success)
   - increment coupon usage counters
   - return success + invoice

   Else:
   - create payment record (failed)
   - return failure reason

```
CREATE TABLE users (
 id INT PRIMARY KEY,
 name TEXT
);
```

```
CREATE TABLE coupons (
 id SERIAL PRIMARY KEY,
 code VARCHAR(50) UNIQUE,
 type VARCHAR(10),          -- 'percentage' or 'fixed'
 value NUMERIC,             -- percent or fixed amount
 max_discount NUMERIC NULL,
 min_order NUMERIC DEFAULT 0,
 expiry DATE,
 active BOOLEAN DEFAULT TRUE,
 usage_limit INT DEFAULT 0,   -- 0 = unlimited
 per_user_limit INT DEFAULT 1
);

CREATE TABLE coupon_usage (
 id SERIAL PRIMARY KEY,
 coupon_id INT REFERENCES coupons(id),
 user_id INT REFERENCES users(id),
 used_count INT DEFAULT 0
);

CREATE TABLE payments (
 id SERIAL PRIMARY KEY,
 user_id INT REFERENCES users(id),
 coupon_id INT NULL REFERENCES coupons(id),
 amount NUMERIC,
 discount NUMERIC,
 final_amount NUMERIC,
 status VARCHAR(20),  -- 'success'|'failed'
 method VARCHAR(50),
 created_at TIMESTAMP DEFAULT now()
```

1.);

Find coupon:

SELECT * FROM coupons WHERE code = 'WELCOME10';

2. Check total usage:

SELECT COALESCE(SUM(used_count),0) AS total_used

FROM coupon_usage WHERE coupon_id = 123;

3. Get user's usage:

SELECT used_count FROM coupon_usage WHERE coupon_id = 123 AND user_id = 45;

4. Record successful payment:

INSERT INTO payments (user_id, coupon_id, amount, discount, final_amount, status, method)

VALUES (45, 123, 1000, 100, 900, 'success', 'card');

5. Increment user coupon usage (insert or update):

6. INSERT INTO coupon_usage (coupon_id, user_id, used_count)
7. VALUES (123,45,1)
8. ON CONFLICT (coupon_id, user_id) DO UPDATE
9. SET used_count = coupon_usage.used_count + 1;
10. **EXAMPLE: Concrete Coupon + Walkthrough**
11. Coupon row:

id	code	type	value	max_discount	min_order	expiry	usage_limit	per_user_limit
123	WELCOME10	percentage	10.00	150.00	500.00	2025-12-31	1000	1

12. User places order:

user_id = 45

- cart_amount = 1200.00
- coupon_code = WELCOME10

Validation:

- cart >= min_order (1200 >= 500) → OK
- percentage 10% → raw discount = 120.0 → below max_discount(150) → discount = 120.00
- final_amount before tax = 1200 - 120 = 1080.00

Assume tax 18% on final_amount (optional):

- tax = 1080 * 0.18 = 194.40
- payable = 1080 + 194.40 = 1274.40

Payment simulation success.

SAMPLE PROGRAM OUTPUT(Console-style)

Applying coupon: WELCOME10

User: 45 Cart: 1200.00

Coupon valid — percentage 10% (max 150.00)

Discount applied: 120.00

Amount after discount: 1080.00

Tax (18%): 194.40

Final payable amount: 1274.40

Payment status: SUCCESS

Payment ID: PAY_20251017_001

Coupon usage updated for user 45 (used_count = 1)

user_id = 45

- coupon_code = WELCOME10

Validation:

- cart >= min_order (1200 >= 500) → OK
- percentage 10% → raw discount = 120.0 → below max_discount(150) → discount = 120.00
- final_amount before tax = 1200 - 120 = 1080.00

Assume tax 18% on final_amount (optional):

- tax = 1080 * 0.18 = 194.40
- payable = 1080 + 194.40 = 1274.40

Payment simulation success.

SAMPLE PROGRAM OUTPUT(Console-style)

Applying coupon: WELCOME10

User: 45 Cart: 1200.00

Coupon valid — percentage 10% (max 150.00)

Discount applied: 120.00

Amount after discount: 1080.00

Tax (18%): 194.40

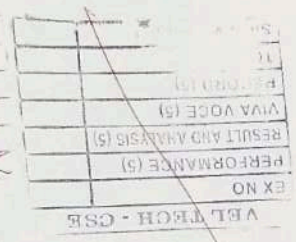
Final payable amount: 1274.40

Payment status: SUCCESS

Payment ID: PAY_20251017_001

Coupon usage updated for user 45 (used_count = 1)

EX NO	11
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	3
RECORD (5)	5
TOTAL (20)	18
SIGN WITH DATE	



Result: 16.10

Thus the Building a Backend Components that 1. Stores Coupons and Payment records.

2. Validates & applies Coupons to orders.
3. Calculates final payable amount and records.