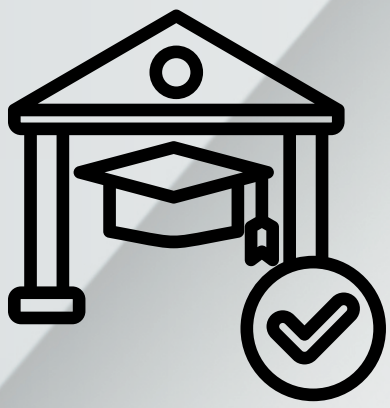


2024 ACPC ADMISSION PROCESS

DBMS PROJECT

THE MAHARAJA SAYAJIRAO UNIVERSITY OF BARODA

सत्यं शिवं सुन्दरम्

Estd. 1949

A PROJECT REPORT
ON
ACPC ADMISSION SYSTEM

COURSE: DATABASE MANAGEMENT SYSTEM (CSE1401)
SUBMITTED BY

NAME	SEATNO	PRN
Vishrut BhadreshBhai Lathiya	486124	8022057517
Darsh Dashrath Patel	486144	8022057511
Vyom Rajendra Shah	486169	8022054043
Kirtan Rajivkumar Soni	486175	8022005012

INDEX

ACPC Admission Process Database Management System

Overview

The ACPC (Admission Committee for Professional Courses) Admission Process Database Management System is designed to streamline and manage the admission process for professional courses in educational institutions. This project aims to develop a robust database system that facilitates efficient management of student data, course information, application processing, merit list generation, and communication with applicants.

Features:

Student Data Management: The system will maintain comprehensive records of student information including personal details, academic qualifications, and contact information.

Course Management: Course details such as available seats, eligibility criteria, syllabus, and faculty information will be stored and managed within the system.

Application Processing: Applicants can submit their applications through an online portal or manually, and the system will process and verify the applications efficiently.

Merit List Generation: Based on predefined criteria such as academic scores, category, and reservation rules, the system will generate merit lists for each course.

Seat Allotment: The system will handle the allocation of seats to eligible candidates based on merit, reservation criteria, and availability of seats in each course. It will ensure fair and transparent seat allotment processes.

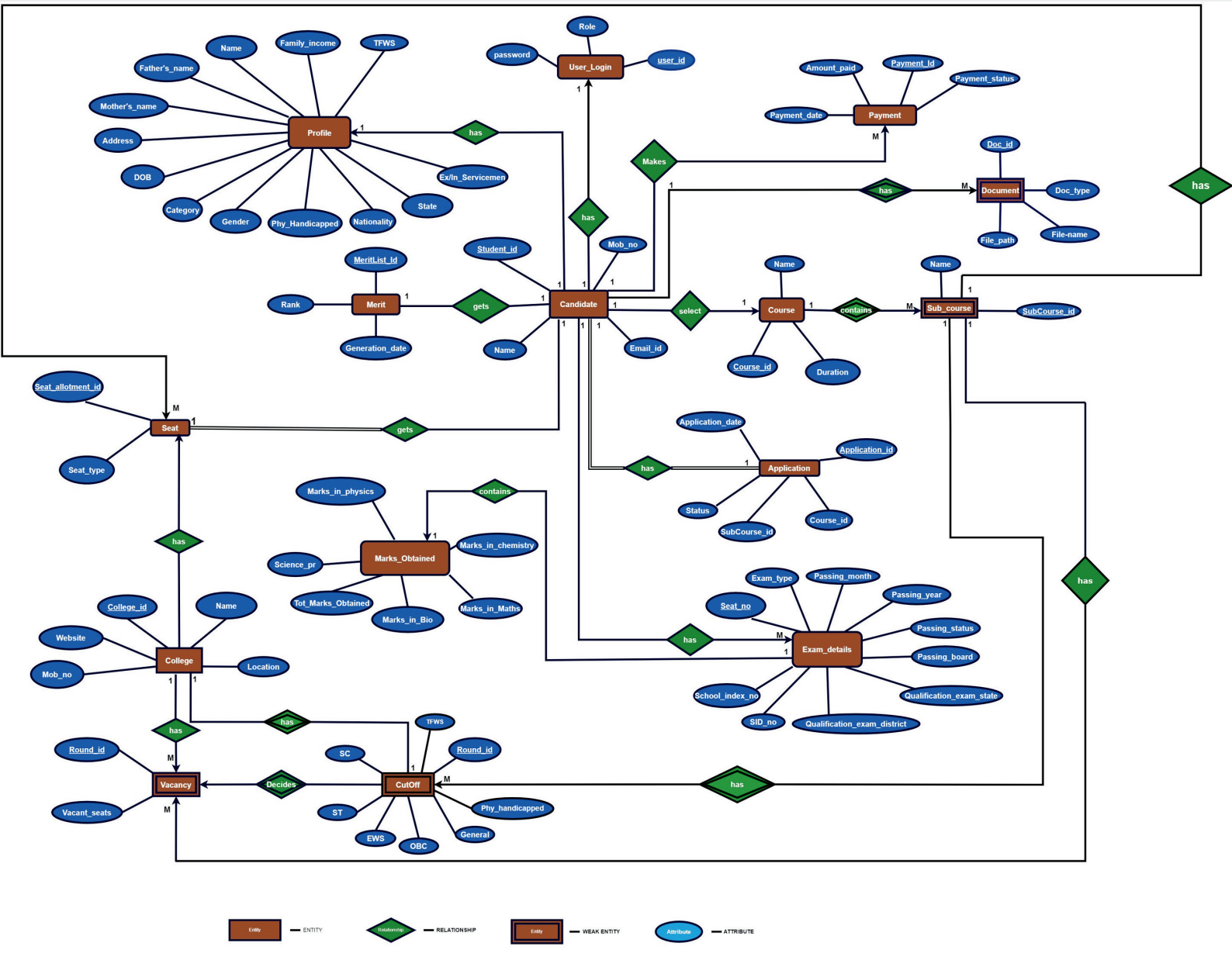
Payment Management: The system will facilitate the collection and management of admission fees and other related payments from applicants. It will provide options for online payment processing and maintain payment records for each student.

.

Assumptions

- If a seat is allotted, then it is assumed that the student has accepted it.
- It is assumed that each college and course has specific cutoff marks for different categories of students (e.g., General, SC, ST).
- It is assumed that exam details and result of all the students has been provided by GSEB to ACPC.
- It is assumed that only 2 rounds of admission have happened.
- It is assumed that there is a verification process for submitted documents to ensure their authenticity.
- It is assumed that there is a specific deadline for submitting applications, and no applications are accepted after this deadline.
- It is assumed that rank is not same even if the candidates are having same science_pr.
- It is assumed that there is a refund policy in place for students who withdraw their admission before a certain date. Refunds may be partial or full, depending on the timing of withdrawal.
- It is assumed that students belonging to reserved categories (e.g., SC, ST, OBC) must provide valid certification of their category during the application process.
- It is assumed that students are allowed to change their course preference after the first round of admission but before the second round. Any seat allotment in the first round will be forfeited if the student opts for a different course.
- It is assumed that cutoff marks may vary between different rounds of admission based on the availability of seats and the performance of applicants in each round.
- By considering these assumptions, we can further refine the database design and develop functionalities to address specific aspects of the admission process.

E R Diagram


Role
password
User_Login
user_id
Name
Family_income
TFWS
Father's_name
Mother's_name
Address
DOB
Category
Gender
Phy_Handicapped
Nationality
State
Ex/In_Servicemen
Profile
has
Amount_paid
Payment_Id
Payment_status
Payment_date
Payment
Makes
Doc_id
Document
Doc_type
File_path
File-name
Student_id
Mob_no
MeritList_Id
Rank
Merit
gets
Candidate
Generation_date
Email_id
select
Course
Course_id
Duration
contains
Sub_course
SubCourse_id
Seat_allotment_id
Seat
Seat_type
Application_date
Application_id
Application
Status
SubCourse_id
Marks_in_physics
Marks_Obtained
Marks_in_chemistry
Science_pr
Tot_Marks_Obtained
Marks_in_Bio
Marks_in_Maths
College_id
College
Website
Mob_no
Location
Exam_type
Passing_month
Passing_year
Seat_no
Passing_status
Passing_board
Exam_details
School_index_no
SID_no
Qualification_exam_district
Qualification_exam_state
Round_id
Vacancy
Vacant_seats
Decides
CutOff
SC
ST
EWS
OBC
Phy_handicapped
General
Entity
ENTITY
Relationship
RELATIONSHIP
WEAK ENTITY
Attribute
ATTRIBUTE

Tables

- **User_login** (**User_id,** Pass, Role)
- **Candidate_Info** (**Student_ID,** Name, Mob_no, Email_id, User_id)
- **Course** (**Course_id,** Name, Duration)
- **SubCourse** (**SubCourse_id,** Name, Course_id)
- **Profile** (**Student_id,** Name, Father's Name, Mother's Name, DOB, Gender, Nationality, State, Category, Phy_Handicapped, Ex/In Servicemen, TFWS, Family_Annual_Income,Perm/Correspondence Address)
- **Cutoff** (**Round_id, Course_id, College_id**, General, EWS, SC, ST, OBC, TFW, Phy_Handicapped)
- **Application** (**App_id,** Student_id, SubCourse_id, App_Date, Status)
- **College** (**College_id,** Name, Loc, Mobile_no, Website)
- **Vacancy** (**College_id, SubCourse_id, Round_id,** Vacant_seats)
- **Exam_Details** (**Seat_no,** Exam_type, Student_id,Passing_month, Passing_year, Passing Status, Passing_board,SID_no)

- **Exams_Details_2**(**Student_id**,Qualification_Exam_State, Qualification_Exam_District,School_Name, School_index_no)
- **Marks_Obtained**(**Seat_no**,Science_pr,Marks_in_physics, Marks_in_chemistry,Marks_in_Maths,Marks_in_Biology, Tot_Marks_Obtained)
- **Payment**(**Payment_id,**Payment_date,Payment_status, Amount_paid, Student_id)
- **Document**(**Doc_id**, File_path,File_name,Doc_type,Student_id)
- **Seat**(**Seat_allotment_id**,Seat_type,Sub_Course_id, Student_id, College_id)
- **Merit** (**Merit_id**, Rank, Generation_date, Student_id)

Conclusion:

The ACPC Admission Process Database Management System aims to modernize and optimize the admission process for professional courses, benefiting both educational institutions and applicants. By leveraging technology to automate and streamline processes, the system enhances efficiency, transparency, and user experience throughout the admission lifecycle.

TRIGGERS

1. TRIGGER TO GENERATE STUDENT ID

->
```
CREATE OR REPLACE TRIGGER GENERATE_STUDENT_ID
BEFORE INSERT ON CANDIDATE_INFO
FOR EACH ROW
DECLARE
  PFX VARCHAR2(2);
  SFX VARCHAR2(4);
  LAST_STUDENT_ID VARCHAR2(8);
BEGIN
  PFX := UPPER(SUBSTR(:NEW.NAME, 1, 2));
  SELECT MAX(TO_NUMBER(SUBSTR(STUDENT_ID,5,4))) INTO LAST_STUDENT_ID
  FROM CANDIDATE_INFO
  WHERE SUBSTR(STUDENT_ID, 3, 2) = '24';
    IF LAST_STUDENT_ID IS NULL THEN
       SFX := '0001';
    ELSE
       SFX := TO_CHAR(TO_NUMBER(SUBSTR(LAST_STUDENT_ID, 5,4)) + 1, 'FM0000');
    END IF;
    :NEW.STUDENT_ID := PFX || '24' || SFX;
END;
```

2.TRIGGER TO GENERATE APPLICATION_ID

->
```
CREATE OR REPLACE TRIGGER GENERATE_APPLICATION_ID
BEFORE INSERT ON APPLICATION
FOR EACH ROW
DECLARE
  APPLICATION_ID_PREFIX VARCHAR2(5) := 'APP_';
  RANDOM_NUMBER VARCHAR2(10);
BEGIN
RANDOM_NUMBER := TO_CHAR(FLOOR(DBMS_RANDOM.VALUE(10000, 99999)));
:NEW.APP_ID := APPLICATION_ID_PREFIX || RANDOM_NUMBER;
END;
```

3.TRIGGER TO CHECK SEATNO

```
-> CREATE OR REPLACE TRIGGER CHECK_SEATNO
   BEFORE INSERT ON EXAM_DETAILS
   FOR EACH ROW
   BEGIN
      IF UPPER(:NEW.EXAM_TYPE)='GUJCET' AND :NEW.SEAT_NO NOT LIKE 'H%' THEN
RAISE_APPLICATION_ERR OR(-20001,'FOR GUJCET EXAM SEAT NUMBER MUST BE START FRFOM
H');
     ELSIF
     UPPER(:NEW.EXAM_TYPE)='HSC' AND :NEW.SEAT_NO NOT LIKE 'B%' THEN
      RAISE_APPLICATION_ERROR(-20001,'SEAT NUMBER MUST BE START FROM B');
     END IF;
   END;
```

4. TRIGGER TO CHECK MOBILE NUMBER

```
-> CREATE OR REPLACE TRIGGER CHECK_MOBILE_LENGTH
   BEFORE INSERT OR UPDATE ON CANDIDATE_INFO
   FOR EACH ROW
   BEGIN
      IF LENGTH(:NEW.MOB_NO) != 10 THEN
         RAISE_APPLICATION_ERROR(-20001, 'MOBILE NUMBER MUST BE 10 DIGITS LONG.');
      END IF;
   END;
```

5. TRIGGER TO GENERATE MERIT ID

```
-> CREATE OR REPLACE TRIGGER GENERATE_MERIT_ID
   BEFORE INSERT ON MERIT
   FOR EACH ROW
   DECLARE
     RANDOM_NUMBER INT;
   BEGIN
     RANDOM_NUMBER := FLOOR(DBMS_RANDOM.VALUE(10000, 99999));
     :NEW.MERIT_ID := RANDOM_NUMBER;
   END;
```

PACKAGE

```
CREATE OR REPLACE PACKAGE P1
AS
FUNCTION GET_PR(A1 IN INT,B OUT FLOAT, C OUT FLOAT)
RETURN FLOAT;
FUNCTION REDUCE_VACANCY(P_COLLEGE_ID IN VARCHAR2, P_SUBCOURSE_ID IN VARCHAR2)
RETURN NUMBER;
PROCEDURE UPDATE_PROFILE(
   P_STUDENT_ID IN PROFILE.STUDENT_ID%TYPE,
   P_NAME IN PROFILE.NAME%TYPE,
   P_FATHER_MOTHER_NAME IN PROFILE.FATHER_MOTHER_NAME%TYPE,
   P_DOB IN PROFILE.DOB%TYPE,
   P_GENDER IN PROFILE.GENDER%TYPE,
   P_NATIONALITY IN PROFILE.NATIONALITY%TYPE,
   P_STATE IN PROFILE.STATE%TYPE,
   P_CATEGORY IN PROFILE.CATEGORY%TYPE,
   P_PHY_HANDICAPPED IN PROFILE.PHY_HANDICAPPED%TYPE,
   P_EX_IN_SERVICEMEN IN PROFILE.EX_IN_SERVICEMEN%TYPE,
   P_TFWS IN PROFILE.TFWS%TYPE,
   P_FAMILY_ANNUAL_INCOME IN PROFILE.FAMILY_ANNUAL_INCOME%TYPE,
   P_ADDRESS IN PROFILE.PERM_CORRESPONDENCE_ADDRESS%TYPE
);
PROCEDURE UPDATE_STATUS( SEAT_ALLOTMENT_ID IN NUMBER);
PROCEDURE RANK_GENERATOR(A IN OUT NUMBER,B IN NUMBER);
PROCEDURE UPDATESUBCOURSE(STUDENT_ID IN NUMBER, SUBCOURSE_ID IN NUMBER,NA IN
VARCHAR2(50));

END P1;
```

FUNCTIONS

1) FUNCTION TO GET AVERAGE PERCENTILE OF BOTH EXAMS:

```
CREATE OR REPLACE FUNCTION GET_PR(ID IN VARCHAR2) RETURN FLOAT IS
  A FLOAT;
  B FLOAT;
  C FLOAT;
BEGIN
  SELECT SCIENCE_PR INTO B FROM MARKS_OBTAINED M
  WHERE M.SEAT_NO = (
    SELECT E.SEAT_NO FROM EXAM_DETAILS E
    WHERE E.SEAT_NO LIKE 'B%' AND E.STUDENT_ID = ID
  );
  SELECT SCIENCE_PR INTO C FROM MARKS_OBTAINED M
  WHERE M.SEAT_NO = (
    SELECT E.SEAT_NO FROM EXAM_DETAILS E
    WHERE E.SEAT_NO LIKE 'H%' AND E.STUDENT_ID = ID
  );
  A := (B + C) / 2;
  RETURN A;
END;
/
```

2) FUNCTION TO REDUCE THE VACANT SEATS:

```
CREATE OR REPLACE FUNCTION REDUCE_VACANCY(P_COLLEGE_ID IN VARCHAR2,
P_SUBCOURSE_ID IN VARCHAR2)
RETURN NUMBER
IS
 V_VACANCY NUMBER;
BEGIN
   SELECT VACANT_SEATS INTO V_VACANCY FROM VACANCY
   WHERE COLLEGE_ID = P_COLLEGE_ID AND SUBCOURSE_ID = P_SUBCOURSE_ID;
IF V_VACANCY > 0 THEN
      UPDATE VACANCY SET VACANT_SEATS = VACANT_SEATS - 1
      WHERE COLLEGE_ID = P_COLLEGE_ID AND SUBCOURSE_ID = P_SUBCOURSE_ID;
      COMMIT;
      RETURN 1;
   ELSE
      RETURN 0;
END IF;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN -1;
   WHEN OTHERS THEN
      RETURN -2;
END;
/
```

PROCEDURES

1) PROCEDURE TO UPDATE THE PROFILE INFORMATION

```
CREATE OR REPLACE PROCEDURE UPDATE_PROFILE(
 P_STUDENT_ID IN PROFILE.STUDENT_ID%TYPE,
 P_NAME IN PROFILE.NAME%TYPE,
 P_FATHER_MOTHER_NAME IN PROFILE.FATHER_MOTHER_NAME%TYPE,
 P_DOB IN PROFILE.DOB%TYPE,
 P_GENDER IN PROFILE.GENDER%TYPE,
 P_NATIONALITY IN PROFILE.NATIONALITY%TYPE,
 P_STATE IN PROFILE.STATE%TYPE,
 P_CATEGORY IN PROFILE.CATEGORY%TYPE,
 P_PHY_HANDICAPPED IN PROFILE.PHY_HANDICAPPED%TYPE,
 P_EX_IN_SERVICEMEN IN PROFILE.EX_IN_SERVICEMEN%TYPE,
 P_TFWS IN PROFILE.TFWS%TYPE,
 P_FAMILY_ANNUAL_INCOME IN PROFILE.FAMILY_ANNUAL_INCOME%TYPE,
 P_ADDRESS IN PROFILE.PERM_CORRESPONDENCE_ADDRESS%TYPE
)

IS
BEGIN
   UPDATE PROFILE
   SET NAME = P_NAME,
      FATHER_MOTHER_NAME = P_FATHER_MOTHER_NAME,
      DOB = P_DOB,
      GENDER = P_GENDER,
      NATIONALITY = P_NATIONALITY,
      STATE = P_STATE,
      CATEGORY = P_CATEGORY,
      PHY_HANDICAPPED = P_PHY_HANDICAPPED,
      EX_IN_SERVICEMEN = P_EX_IN_SERVICEMEN,
      TFWS = P_TFWS,
      FAMILY_ANNUAL_INCOME = P_FAMILY_ANNUAL_INCOME,
      PERM_CORRESPONDENCE_ADDRESS = P_ADDRESS
   WHERE STUDENT_ID = P_STUDENT_ID;

   COMMIT;
END UPDATE_PROFILE;
```

2) PROCEDURE TO GENERATE THE MERIT RANK

```
CREATE OR REPLACE PROCEDURE RANK_GENERATOR(A IN OUT NUMBER,ID IN
VARCHAR)
IS
BEGIN
   INSERT INTO MERIT(RANK,GENERATION_DATE,STUDENT_ID) VALUES
(A,SYSDATE,ID);
   A := A + 1;
END
```

3)PROCEDURE TO UPDATE THE STATUS OF SEAT ALLOTED

```
CREATE OR REPLACE PROCEDURE UPDATE_STATUS(SEAT_ALLOTMENT_ID IN
NUMBER)
IS
BEGIN
      UPDATE APPLICATION
      SET STATUS = 'ALLOTED'
      WHERE APP_ID = (
         SELECT A.APP_ID
         FROM APPLICATION A
         JOIN SEAT S ON A.STUDENT_ID = S.STUDENT_ID
         WHERE S.SEAT_ALLOTMENT_ID = SEAT_ALLOTMENT_ID
      );
EXCEPTION
      WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('AN UNEXPECTED ERROR OCCURRED: ' ||
SQLERRM);
END UPDATE_STATUS;
END P1;
```

4)PROCEDURE FOR CHANGING THE SUBCOURSE

```
CREATE OR REPLACE PROCEDURE UPDATESUBCOURSE(
 STUDENT_ID IN NUMBER,SUBCOURSE_ID IN NUMBER,NA IN
VARCHAR(50)
) IS
N NUMBER;
BEGIN

SELECT SUBCOURSE_ID INTO N FROM SUBCOURSE
WHERE NAME=NA;
   UPDATE APPLICATION SET SUBCOURSE_ID = N;
EXCEPTION
   WHEN OTHERS THEN
      -- ERROR HANDLING
      DBMS_OUTPUT.PUT_LINE('ERROR OCCURRED: ' || SQLERRM);
END;
/
```

DDLs

```
CREATE TABLE User_login (
    User_id INT PRIMARY KEY,
    Pass VARCHAR(255) NOT NULL,
    Role VARCHAR(50) NOT NULL
);
CREATE TABLE Candidate_Info (
    Student_ID Varchar(50) PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Mob_no VARCHAR(15),
    Email_id VARCHAR(100),
    User_id INT,
    FOREIGN KEY (User_id) REFERENCES User_login(User_id)
);
CREATE TABLE Course (
    Course_id INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Duration INT
);
CREATE TABLE SubCourse (
    SubCourse_id INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Course_id INT,
    FOREIGN KEY (Course_id) REFERENCES Course(Course_id)
);

CREATE TABLE Application (
    App_id VARCHAR(50) PRIMARY KEY,
    Student_id VARCHAR(50),
    SubCourse_id INT,
    App_Date DATE,
    Status VARCHAR(50),
    FOREIGN KEY (Student_id) REFERENCES Candidate_Info(Student_ID),
    FOREIGN KEY (SubCourse_id) REFERENCES SubCourse(SubCourse_id)
);
```

```
CREATE TABLE Profile (
    Student_id VARCHAR(50) PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Father_Mother_Name VARCHAR(100),
    DOB DATE,
    Gender VARCHAR(10),
    Nationality VARCHAR(50),
    State1 VARCHAR(50),
    Category VARCHAR(50),
    Phy_Handicapped VARCHAR(50) CHECK(Phy_Handicapped IN('YES','NO')),
    Ex_In_Servicemen VARCHAR(50) CHECK ( Ex_In_Servicemen IN('YES','NO')),
    TFWS VARCHAR(50) CHECK (TFWS IN('YES','NO')) ,
    Family_Annual_Income NUMBER(10, 2),
    Address VARCHAR(255),
    foreign key (Student_id) references Candidate_Info(Student_id)
);
CREATE TABLE Cutoff (
    Round_id INT,
    Sub_Course_id INT,
    College_id INT,
    General INT,
    EWS INT,
    SC INT,
    ST INT,
    OBC INT,
    TFW INT,
    Phy_Handicapped INT,
    PRIMARY KEY (Round_id, Course_id, College_id)
    FOREIGN KEY (SubCourse_id) REFERENCES SubCourse(SubCourse_id),
    FOREIGN KEY (College_id) REFERENCES College(College_id)
);
CREATE TABLE College (
    College_id INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Loc VARCHAR(100),
    Mobile_no NUMBER,
    Website VARCHAR(255)
);
```

```
CREATE TABLE Vacancy (
    College_id INT,
    SubCourse_id INT,
    Round_id INT,
    Vacant_seats INT,
    PRIMARY KEY (College_id, SubCourse_id, Round_id),
    FOREIGN KEY (College_id,SubCourse_id,Round_id) REFERENCES
Cutoff(College_id,SubCourse_id,Round_id)
);
CREATE TABLE Exam_Details (
    Exam_type VARCHAR(100),
    Seat_no VARCHAR2(10),
    Student_id VARCHAR(50)
    Passing_month VARCHAR(20),
    Passing_year INT,
    Passing_Status VARCHAR(50),
    Passing_board VARCHAR(100),
    SID_no VARCHAR(50),
    Foreign key(Student_id) references Candidate_Info(Student_id),
    CONSTRAINT pK_1 PRIMARY KEY(Exam_type, Seat_no)
);
CREATE TABLE Exams_Details_2 (
    Student_id VARCHAR(50) PRIMARY KEY,
    Qualification_Exam_State VARCHAR(100),
    Qualification_Exam_District VARCHAR(100),
    School_Name VARCHAR(100),
    School_index_no VARCHAR(50),
    Foreign key(Student_id) references Candidate_Info(Student_id)
);
CREATE TABLE Merit (
    Merit_id INT PRIMARY KEY,
    Rank INT,
    Generation_date DATE,
    Student_id VARCHAR(50),
    FOREIGN KEY (Student_id) REFERENCES Candidate_Info(Student_ID)
);
```

```
CREATE TABLE Marks_Obtained (
    Seat_no VARCHAR2(10) PRIMARY KEY,
    Science_pr DECIMAL(5, 2),
    Marks_in_physics DECIMAL(5, 2),
    Marks_in_chemistry DECIMAL(5, 2),
    Marks_in_Maths DECIMAL(5, 2),
    Marks_in_Biology DECIMAL(5, 2),
    Tot_Marks_Obtained DECIMAL(5, 2),
    FOREIGN KEY (Seat_no) REFERENCES Exam_details(Seat_no)
);
CREATE TABLE Payment (
    Payment_id INT PRIMARY KEY,
    Payment_date DATE,
    Payment_status VARCHAR(50),
    Amount_paid DECIMAL(10, 2),
    Student_id VARCHAR(50),
    FOREIGN KEY (Student_id) REFERENCES Candidate_Info(Student_ID)
);
CREATE TABLE Document (
    Doc_id INT PRIMARY KEY,
    File_path VARCHAR(255),
    File_name VARCHAR(100),
    Doc_type VARCHAR(50),
    Student_id VARCHAR(50),
    FOREIGN KEY (Student_id) REFERENCES Candidate_Info(Student_ID)
);
CREATE TABLE Seat (
    Seat_allotment_id INT PRIMARY KEY,
    Seat_type VARCHAR(50),
    Sub_Course_id INT,
    Student_id VARCHAR(50),
    College_id INT,
    FOREIGN KEY (Sub_Course_id) REFERENCES SubCourse(SubCourse_id),
    FOREIGN KEY (Student_id) REFERENCES Candidate_Info(Student_ID),
    FOREIGN KEY (College_id) REFERENCES College(College_id)
);
```