Practical 1:

#Step1. Create Database:

```
CREATE DATABASE IF NOT EXISTS PRACTICAL;

USE PRACTICAL;
```

 # Step2. CREATE TABLE

```
CREATE TABLE IF NOT EXISTS STUDENTS
(    S_id int primary key,
     S_Name varchar(55),
     DOB date,
     City varchar(25));

describe students;
```

 # ALTER attributes of the table

```
alter table students modify s_name varchar(55) not null;

ALTER TABLE studentS MODIFY DOB date not null;

ALTER TABLE studentS MODIFY CITY varchar(25) not null;

ALTER TABLE STUDENTS ADD COLUMN EMAIL varchar(60) UNIQUE;

ALTER TABLE STUDENTS ADD COLUMN Gender char(1);

ALTER TABLE STUDENTS MODIFY COLUMN GENDER CHAR(1) AFTER S_NAME;
```

 # insert function

```
INSERT INTO students values

(1001, "Hari Singh", "M", "2002/05/25",
"Kathmandu","singh_hari@gmail.com");


INSERT INTO students (s_ID, S_Name, gender)
```

```
 values   (1002, "Janak Singh", "M"),
          (1003, "Sita Karki", "F");


INSERT INTO students values
(1004, "Jamuna Shahi", "F", "2001/10/28", "Butwal",
"jamunashahi1@gmail.com"),
 (1005, "Sajan KC", "M", "2000/02/02", "Bhaktapur",
"kc_sajan@gmail.com"),
 (1006, "Ram Thapa", "M", "2010/06/12", "Pokhara", "null");
```

#update records

```
 UPDATE students
 SET city="Kathmandu"
 WHERE s_ID=1003;
```

```
  select *from students;   # DISPLAY ALL RECORDS
```

#Display specified columns

```
  SELECT S_id, S_name, Email from Students;
```

 # WHERE Clause For Filter data

```
  SELECT * FROM Students
  WHERE CITY="kATHMANDU";


 SELECT * FROM Students
  WHERE CITY="kATHMANDU" AND GENDER='M';


 SELECT * FROM Students
```

```
 WHERE CITY="kATHMANDU" OR CITY="POKHARA";

SELECT * FROM Students
 WHERE NOT CITY="kATHMANDU";

SELECT *FROM students WHERE s_name LIKE 'H%';
```

 # Group by clause

```
SELECT CITY, COUNT(S_ID) AS TOTAL_STUDENTS
FROM students GROUP BY CITY;
```

 # HAVING clause

```
SELECT CITY, COUNT(s_ID) AS TOTAL_STUDENTS
FROM studentS group by CITY HAVING COUNT(S_id)<3;
```

 # ORDER by

```
SELECT * FROM students ORDER BY S_NAME;

 SELECT * FROM students ORDER BY S_NAME DESC;
```

 # STRING FUNCTION: UPPER/UCASE, LOWER/ LCASE

```
 SELECT UCASE('bca program');

 SELECT LCASE("BCA PROGRAM");

 SELECT character_length('BCA PROGRAM'); # or char_lenght()

 SELECT S_name, character_length(S_NAME) AS NAME_LENGHT
```

```
  FROM students;

  select concat('mr. ',"Raj") as join_name;

  SELECT s_ID, concat("MR/MS ",s_name,"-",city) as togather
  from students;

  select reverse('nepal');
```

#replace function

```
  select replace("apple is veg", "veg","fruit");
```

 # trim functions: ltrim

```
  select length("   NEPAL      ");
  select length(LTRIM("   NEPAL      "));
  select Length(RTRIM("   NEPAL      "));
select length(TRIM("   NEPAL      "));
```

 # Number functions:

```
select abs(-25);
select mod(5,2);
select mod(25,4) as remainder;
select power(3,2) as power;
select sqrt(81);
select greatest(55,24,42,64,22);
select least(55,24,42,64,22);
select truncate(2.5378,2);
```

 # truncate function defined the number of digits after decimal

```
select round(5.86); #find the round ceiling number;

select round(2.5678,2);

select ceil(5.8);

select floor(5.8);

select sum(3+5+3+8);
```

Practical 2:

Pg. 158, Q 15. Create the following table by specifying the Primary key, Not NULL, foreign key constraints DDL statement in SQL:

1. **Department (dept_no, d_name, city),**
2. **Employee (emp_id, e_name, salary) and**
3. **Works (dept_no, emp_id)**

```
CREATE DATABASE IF NOT EXISTS q15;

USE Q15;

-- Create the Department table

CREATE TABLE Department (

    dept_no INT PRIMARY KEY,          -- Primary Key constraint

    d_name VARCHAR(50) NOT NULL,      -- Not NULL constraint

    city VARCHAR(50) NOT NULL         -- Not NULL constraint

);


-- Create the Employee table

CREATE TABLE Employee (

    emp_id INT PRIMARY KEY,           -- Primary Key constraint

    e_name VARCHAR(50) NOT NULL,      -- Not NULL constraint
```

```
    salary DECIMAL(10, 2) NOT NULL DEFAULT 40000    -- Not NULL
constraint

);
```

-- Create the Works table

```
CREATE TABLE Works (

    dept_no INT NOT NULL,             -- Foreign Key to
Department.dept_no

    emp_id INT NOT NULL,              -- Foreign Key to
Employee.emp_id

    PRIMARY KEY (dept_no, emp_id),    -- Composite Primary Key
constraint

    FOREIGN KEY (dept_no) REFERENCES Department(dept_no)

        ON DELETE CASCADE ON UPDATE CASCADE,  -- Foreign Key
constraint with cascading options

    FOREIGN KEY (emp_id) REFERENCES Employee(emp_id)

        ON DELETE CASCADE ON UPDATE CASCADE   -- Foreign Key
constraint with cascading options

);
```

Note: 🎬 A composite primary key is defined on dept_no and emp_id to ensure unique combinations.

🎬 Foreign keys:

- dept_no references Department(dept_no).
- emp_id references Employee(emp_id).

🎬 ON DELETE CASCADE ON UPDATE CASCADE ensures related rows are maintained or deleted appropriately.

Practical 3:

Consider the following schemas: BOOK (Book_ID, Title, Publisher_ID, Year_of Pub, Price) AUTHOR (Author_ID, Book_ID, Author Name) PUBLISHER

(Publisher_ID, Book_ID, Address, Name_of Pub, No._of Copies) Write a query in SQL for the following:

(i) **to create a table and insert some data and Find the name of authors whose books are published by "ABC Press".**

(ii) **Find the name of the author and price of the book, whose Book_ID is '100'.**

(iii) **Find the title of the books which are published by Publisher_ID '20' and are published in 2011.**

(iv) **Find the address of the publisher who has published Book_ID "500".**

```
CREATE DATABASE P1;

USE P1;
```

-- Create BOOK Table

```
CREATE TABLE BOOK (
    Book_ID INT PRIMARY KEY,
    Title VARCHAR(100) NOT NULL,
    Publisher_ID INT NOT NULL,
    Year_of_Pub INT NOT NULL,
    Price DECIMAL(10, 2) NOT NULL
);
```

-- Create AUTHOR Table

```
CREATE TABLE AUTHOR (
    Author_ID INT PRIMARY KEY,
    Book_ID INT NOT NULL,
    Author_Name VARCHAR(100) NOT NULL,
    FOREIGN KEY (Book_ID) REFERENCES BOOK(Book_ID)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

-- Create PUBLISHER Table

```
CREATE TABLE PUBLISHER (
```

```
    Publisher_ID INT PRIMARY KEY,
    Book_ID INT NOT NULL,
    Address VARCHAR(255) NOT NULL,
    Name_of_Pub VARCHAR(100) NOT NULL,
    No_of_Copies INT NOT NULL,
    FOREIGN KEY (Book_ID) REFERENCES BOOK(Book_ID)
        ON DELETE CASCADE ON UPDATE CASCADE
);

-- Insert Sample Data
INSERT INTO BOOK (Book_ID, Title, Publisher_ID, Year_of_Pub, Price)
VALUES
    (100, 'Book One', 20, 2011, 299.99),
    (200, 'Book Two', 30, 2020, 499.99),
    (500, 'Book Three', 20, 2015, 399.99);

INSERT INTO AUTHOR (Author_ID, Book_ID, Author_Name)
VALUES
    (1, 100, 'John Doe'),
    (2, 200, 'Jane Smith'),
    (3, 500, 'Alice Brown');

INSERT INTO PUBLISHER (Publisher_ID, Book_ID, Address, Name_of_Pub,
No_of_Copies)
VALUES
    (20, 100, '123 Main St', 'ABC Press', 500),
    (30, 200, '456 Elm St', 'XYZ Publishers', 300),
    (10, 500, '123 Main St', 'ABC Press', 700);
```

```
select * from publisher;
```

-- Find Authors Published by "ABC Press":

```
SELECT DISTINCT A.Author_Name
FROM AUTHOR A
JOIN BOOK B ON A.Book_ID = B.Book_ID
JOIN PUBLISHER P ON B.Publisher_ID = P.Publisher_ID
WHERE P.Name_of_Pub = 'ABC Press';
```

-- (ii) Find the Name of the Author and Price of the Book with Book_ID = 100

```
SELECT A.Author_Name, B.Price
FROM AUTHOR A
JOIN BOOK B ON A.Book_ID = B.Book_ID
WHERE B.Book_ID = 100;
```

-- (iii) Find Titles of Books Published by Publisher_ID = 20 in 2011

```
SELECT B.Title
FROM BOOK B
WHERE B.Publisher_ID = 20 AND B.Year_of_Pub = 2011;
```

-- (iv) Find Address of Publisher Who Published Book_ID = 500

```
SELECT P.Address
FROM PUBLISHER P
WHERE P.Book_ID = 500;
```

Domain Types in SQL

1. **Char (n).** Fixed length character string, with user-specified length n.
2. **Varchar(n).** Variable length character strings, with user-specified maximum length n.
3. **int.** Integer (a finite subset of the integers that is machine-dependent).
4. **smallint.** Small integer (a machine-dependent subset of the integer domain type).
5. **Numeric (p,d).** Fixed point number, with user-specified precision of p digits, with n digits to the right of decimal point.
6. **Real, double precision.** Floating point and double-precision floating point numbers, with machine-dependent precision.
7. **Float (n).** Floating point number, with user-specified precision of at least n digits.
8. **date:** Dates, containing a (4 digit) year, month and date
    a. Example: **date** '2024-7-27'
9. **Time:** Time of day, in hours, minutes and seconds.
    a. Example: **time** '09:00:30' **time** '09:00:30.75'
10. **timestamp**: date plus time of day
    a. Example: **timestamp** '2024-7-27 09:00:30.75'
11. **interval:** period of time
    a. Example: interval '1' day
    b. Subtracting a date/time/timestamp value from another gives an interval value
    c. Interval values can be added to date/time/timestamp values
12. **Operations on complex types:**
    a. Subtracting a date/time/timestamp value from another gives an interval value
    b. Interval values can be added to date/time/timestamp values
    c. Values of individual fields can be extracted from date/time/timestamp: **extract** (**year from** r.starttime)
    d. String types can typically be cast to date/time/timestamp: **cast** <string-valued- expression> **as date**

Integrity constraints

Integrity constraints ensure that changes made to the database by authorized users do not result in loss of data consistency.

Domain Constraints

1. A domain of possible values must be associated with every attribute in the database.
2. Declaring an attribute of a particular domain acts as a restraint on the values it can take.
3. They are easily tested by the system. EX1: cannot set an integer variable to "cat".

Data Integrity

1. Databases are used to store data

2. The data is used to create information which is needed for making decisions. Therefore, we need to make sure that the data which is stored in the database is correct and consistent. This is known as **data integrity**.
3. For relational databases, there are **entity integrity** and **referential integrity** rules which help to make sure that we have data integrity

 # Attribute Integrity

Attribute integrity is not part of the relational model. It is used by database software to help with data integrity. The software makes sure that data for particular fields is of the correct type (eg letters or numbers) or the correct length

 # Entity Integrity

1. The entity integrity rule applies to Primary Keys. The entity integrity rule says that the value of a Primary Key in a table must be unique and it can never have no value (null)
2. Operations on the database which insert new data, update existing data, or delete data must follow this rule

 # Referential Integrity

1. The referential integrity key applies to Foreign Keys. A relation schema may have an attribute that corresponds to the primary key of another relation. The attribute is called a **foreign key.**
2. The referential integrity key says that the value of a Foreign key must either be null (ie have no value) or be equal to the value in the linked table where the Foreign Key is the Primary Key
3. Ensuring that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation.
4. EX1: In a banking system, the attribute branch-name in Account-Schema is a foreign key referencing the primary key of Branch-Schema.

 # Database Modification

1. Inserting, deleting and updating can cause violations of referential integrity.
2. Therefore, the system must check that referential integrity is maintained when you perform these operations.
3. If referential integrity is violated during these operations, the default action is to reject the operation.
4. Referential Integrity in SQL: Foreign Keys
5. Foreign Keys are specified as part of the SQL ‘create table’ statement by using the ‘foreign key’ clause.
6. By default, foreign key references the primary key attributes of the referenced table.

Cascading

- ➔ When referential integrity is violated during a modification, instead of just rejecting the modification, you can cascade:
  - ◆ Delete cascade
  - ◆ Update cascade
- ➔ Delete Cascade
  - ◆ In a delete cascade, anything that has references to the deleted item is also deleted.
- ➔ Update Cascade
  - ◆ In an update cascade, when the updated item results in a violation of referential integrity, the system will update accordingly to fix the problem.

Integrity Constraints in Create Table

Constraint restricts the values that the table can store. We can declare integrity constraints at the table level or column level. In column-level constraints the constraint type is specified after specifying the column data type i.e., before the delimiting comma. Whereas in the table-level constraints the constraint type is going to be specified as separate comma-delimited clauses after defining the columns.

There are six constraints

1. Not Null
2. Unique Key
3. Check
4. Primary Key
5. Foreign Key
6. Default

1. Not Null

If a column in a table is specified as Not Null, then it's not possible to insert a null in such a column. It can be implemented with create and alter commands. When we implement the Not Null constraint with the alter command there should not be any null values in the existing table.

2. Unique Key

- ➔ The unique constraint doesn't allow duplicate values in a column. If the unique constraint encompasses two or more columns, no two equal combinations are allowed. Note: There is a different behavior in the following Relational Databases.
- ➔ MS SQL Server: In this, we can insert one Row with a Null value against the Unique Key constraint column.

- ➔ Oracle: In this, we can insert any number of Rows with a Null value against the Unique Key constraint column. Please keep it in mind that two Null values are not equal.

3. Check

Check constraint is used to restrict the values before inserting into a table.

4. Primary Key

- ➔ The key column with which we can identify the entire Table is called as a primary key column. A primary key is a combination of a Unique and a Not Null constraint; it will not allow null and duplicate values. A table can have only one primary key.
- ➔ A primary key can be declared on two or more columns as a Composite Primary Key.

5. Foreign Key

Columns defined as foreign keys refer the Primary Key of other tables. The Foreign Key "points" to a primary key of another table, guaranteeing that you can t enter data into a table unless the referenced table has the data already which enforces the REFERENTIAL INTEGRITY. This column will take Null values.

6. Default

The DEFAULT constraint is used to insert a default value into a column.
The default value will be added to all new records, if no other value is specified.

Data Definition Language (DDL)

TABLE CREATION Syntax

CREATE TABLE <table name> (
<column_name1> <data type>[(<width>)]
[constraint <constraint name> <constraint type>],
<column_name2> <data type>[(<width>)],
<column_name3> <data type>[(<width>)],
<column_name4> <data type>[(<width>)],
.....
<column_nameN> <data type>[(<width>)]
);

Example

Create table section **(** course_id varchar (8),
sec_id varchar (8) , Semester varchar (6),

```
        Year numeric (4,0),
        Building varchar (15),
        room_number varchar (7),
        Time slot id varchar (4),
        Primary key (course_id, sec_id, semester, year),
        Check (semester in (’Fall’, ’Winter’, ’Spring’, ’Summer’))

Create table person (   ID char (10),
        name char(40),
        mother char(10),
        father char(10),
        primary key ( ID),
        foreign key father references person,
        foreign key mother references person )

  Create table course (
      course_id char(5) primary key,
      title  varchar(20),
      dept_name varchar(20)
      foreign key (dept_name) references department on delete cascade on update
                        cascade )

   Create table Employee (
          mpno number (4) constraint pk_emp primary key,
          ename varchar2(50),
          salary number(10,2),
          hire_date date,
          gender char(1) constraint chk_gen check(gender in ('M', 'F', 'm', 'f')),
          email varchar2(50) unique );

CREATE table Persons (
        P_Id int NOT NULL,
        LastName varchar(255) NOT NULL,
        FirstName varchar(255),
        Address varchar(255),
        City varchar(255) DEFAULT 'Sandnes' )
```