Name	**Jay parmar**
Endrollment No -	2202031000079
Subject	**Ojt** **Lab-2**
Course	**B.TECH(IT)**

OJT Program

Write a C program to print the address of a variable using a pointer.?

```
#include <stdio.h>

int main() {
int num = 42;
int *ptr = #



printf("The address of 'num' is: %p\\n", &num); printf("The value of 'ptr' is: %p\\n",
ptr);  printf("The value of '*ptr' is: %d\\n", *ptr);  return 0;


}

// Output
The address of 'num' is: 0x7ffcb3c13b2c
The value of 'ptr' is: 0x7ffcb3c13b2c
The value of '*ptr' is: 42
```

Write a C program to create a Calculator using a pointer.

```
#include <stdio.h>

int main() {
    double num1,
   num2;  char op;
   double *result;

    printf("Enter two numbers and an operator (+, -, *, /): ");
   scanf("%lf %lf %c", &num1, &num2, &op);

    switch(op) {
      case '+':
          *result = num1 + num2;
         break;
       case '-':
          *result = num1 - num2;
         break;
       case '*':
```

```
            *result = num1 * num2;
            break;
        case '/':
            *result = num1 / num2;
            break;
        default:
            printf("Invalid operator");
            return 1;
    }

    printf("The result is: %lf", *result);

    return 0;
}

//output
Enter two numbers and an operator (+, -, *, /): 5.6 2.3 *
The result is: 12.880000
```

Write a C program to swap the two values using call by value and call by reference.

```
#include <stdio.h>

void swap_by_value(int x, int y)
  {  int temp = x;  x = y;  y =
  temp;
}

void swap_by_reference(int *x, int *y)
  {  int temp = *x;  *x = *y;
   *y = temp;
}

int main() {  int a
  = 5, b = 7;

    // Call swap_by_value
    printf("Before swap_by_value: a = %d, b = %d\\n", a,
  b);  swap_by_value(a, b);  printf("After swap_by_value:
  a = %d, b = %d\\n", a, b);

    // Call swap_by_reference  printf("Before
  swap_by_reference: a = %d, b = %d\\n", a, b);
```

```
    swap_by_reference(&a, &b);  printf("After
    swap_by_reference: a = %d, b = %d\\n", a, b);

    return 0;
}
// output
Before swap_by_value: a = 5, b = 7
After swap_by_value: a = 5, b = 7
Before swap_by_reference: a = 5, b = 7
After swap_by_reference: a = 7, b = 5
```

Define a structure type struct personal that would contain person name, Date of birth and age?

```
#include <stdio.h>

// Define the struct  struct personal {  char name[50];  char dob[11];  // Assuming date of birth
will be stored as a string in the format "MM/DD/YYYY"  int age;
};

int main() {
    // Create an instance of the struct
    struct personal person1;

    // Initialize the struct fields  printf("Enter person's name: ");
    scanf("%s", person1.name);  printf("Enter person's date of
    birth (in MM/DD/YYYY format): ");  scanf("%s", person1.dob);
    printf("Enter person's age: ");  scanf("%d", &person1.age);

    // Print out the struct fields  printf("Person's name:
    %s\\n", person1.name);  printf("Person's date of
    birth: %s\\n", person1.dob);  printf("Person's age:
    %d\\n", person1.age);

    return 0;
}
// output
Enter person's name: John Smith
Enter person's date of birth (in MM/DD/YYYY format): 01/01/1990
Enter person's age: 33
Person's name: John Smith
Person's date of birth: 01/01/1990
Person's age: 33
```

Write a C program to calculate the sum of n numbers entered by the user using dynamic memory allocation.

```
#include <stdio.h>
#include <stdlib.h>

int main() {  int n,
   i, sum = 0;  int*
   arr;

    // Get the number of elements from the user
   printf("Enter the number of elements: ");
   scanf("%d", &n);

    // Allocate memory dynamically for the array
   arr = (int*)malloc(n * sizeof(int));

    // Read in the elements from the
   user  printf("Enter the %d
   elements:\\n", n);  for (i = 0; i < n;
   i++) {  scanf("%d", &arr[i]);
    }

    // Calculate the sum of the
   elements  for (i = 0; i < n; i++) {
   sum += arr[i];
    }

    // Print out the sum
   printf("Sum = %d\\n", sum);

    // Free the dynamically allocated memory
    free(arr);

    return 0;
}
// output
Enter the number of elements: 5
Enter the 5 elements:
1 2 3 4 5
Sum = 15
```

Write a C program to calculate the sum of n numbers entered by the user using dynamic

memory allocation

```
#include <stdio.h>
#include <stdlib.h>

int main() {  int n,
   i, sum = 0;  int*
   arr;

    // Get the number of elements from the user
   printf("Enter the number of elements: ");
   scanf("%d", &n);

    // Allocate memory dynamically for the array
   arr = (int*)malloc(n * sizeof(int));

    // Read in the elements from the
   user  printf("Enter the %d
   elements:\\n", n);  for (i = 0; i < n;
   i++) {  scanf("%d", &arr[i]);
    }

    // Calculate the sum of the
   elements  for (i = 0; i < n; i++) {
   sum += arr[i];
    }

    // Print out the sum
   printf("Sum = %d\\n", sum);  //
   Free the dynamically allocated
   memory
    free(arr);

    return 0;
}
// output

Enter the number of elements: 5
Enter the 5 elements:
1 2 3 4 5
Sum = 15
```

Write a C++ program that prompts the user to enter a letter and check whether a letter is a vowel or constant?

```
#include <iostream>

#include <cctype>

using namespace std;

int main() {  char ch;  cout
   << "Enter a letter: ";  cin
   >> ch;

    // Convert the letter to lowercase for easier comparison
    ch = tolower(ch);

    if (ch >= 'a' && ch <= 'z') {  if (ch == 'a' || ch == 'e' || ch ==
       'i' || ch == 'o' || ch == 'u') {  cout << ch << " is a vowel."
       << endl;
        } else {
            cout << ch << " is a consonant." << endl;
        }
    } else {  cout << "Invalid input. Please enter a letter from a to z."
       << endl;
    }

    return 0;
}
// output  Enter
a letter: a  a is
a vowel.  Enter
a letter: b  b is
a consonant.
Enter a letter: 1
Invalid input. Please enter a letter from a to z.
```

Write a C++ program to demonstrate the concept of constructor and destructor?

```
#include <iostream>

using namespace std;

class MyClass {
public:
```

```
    // Constructor
    MyClass() {  cout << "Constructor
       called." << endl;
    }

    // Destructor
    ~MyClass() {  cout << "Destructor
       called." << endl;
    }
};

int main() {  cout << "Creating
   object." << endl;  MyClass obj;

    cout << "Object created." << endl;
   return 0;
}
// output
Creating object.
Constructor called.
Object created.
Destructor called.
```

Write a C++ program to implement Multilevel Inheritance.?

```
#include <iostream>
using namespace std;

// Base class
class Animal {
public:
    void eat() {
        cout << "I can eat." << endl;
    }
};

// Intermediate class  class
Mammal : public Animal {
public:
    void run() {
        cout << "I can run." << endl;
    }
};
```

```
// Derived class  class Cat :
public Mammal {  public:
    void meow() {
        cout << "I can meow." << endl;
    }
};

int main() {
    // Create a Cat object
    Cat cat;

    // Call methods from all
    classes  cat.eat();  cat.run();
    cat.meow();

    return 0;
}

// output
I can eat.
I can run.
I can meow.
```

Write a C++ program to overload binary + operator.?

```
#include <iostream>
using namespace std;  //
Define a class for
complex numbers  class
Complex {  private:
    double real;
double imaginary;
public:
    Complex(double r = 0, double i = 0)
      {  real = r;  imaginary = i;
    }

    // Overload the + operator
    Complex operator +(const Complex& obj) {
      Complex res;  res.real = real + obj.real;
      res.imaginary = imaginary +
      obj.imaginary;  return res;
    }
```

```
    void display() {  cout << real << " + " << imaginary
       << "i" << endl;
    }
};

int main() {
    // Create two complex numbers
    Complex num1(2, 3);
    Complex num2(4, 5);

    // Add them using the overloaded + operator
    Complex sum = num1 + num2;

    // Display the result
    sum.display();

    return 0;
}
// output
6 + 8i
```

Write a C++ program to understand the concept of run time polymorphism?

```
#include <iostream>
using namespace std;

// Base class  class
Animal {  public:  //
Virtual method  virtual
void sound() {
        cout << "The animal makes a sound." << endl;
    }
};

// Derived class  class
Dog : public Animal {
public:
    // Override the virtual method
    void sound() {
        cout << "The dog barks." << endl;
    }
};
```

```
int main() {
    // Create an Animal pointer and a Dog object
    Animal* animal;
    Dog dog;

    // Assign the Dog object to the Animal pointer
   animal = &dog;

    // Call the virtual method using the pointer  animal-
   >sound();

    return 0;
}
// output
The dog barks.
```

Make a Resume using the HTML tags without CSS.?

```
<!DOCTYPE html>
<html>
<head>
        <title>Resume</title>
</head>
<body>
        <h1>Pavandeep kumar</h1>
        <h3>Contact Information</h3>
        <p>Address: silver oak university hmedabad c block hostel</p>
        <p>Phone: 9798949232</p>
        <p>Email: pavandeepkumarmlk@email.com</p>
        <h3>Summary</h3>
        <p>Experienced software engineer with a focus on web development and database
management. Proven ability to design and implement scalable and reliable software solutions.
Strong problem-solving and analytical skills.</p>
        <h3>Experience</h3>
        <h4>Software Engineer, Devstree it service pvt ltd</h4>
        <p>Developed and maintained web applications using HTML, CSS, JavaScript, and
 PHP.</p>
        <p>Designed and implemented database schemas and queries using MySQL.</p>
        <p>Collaborated with cross-functional teams to deliver high-quality software
products.</p>
        <h4>Web Developer, ABC Company</h4>
        <p>Developed and maintained client websites using HTML, CSS, and JavaScript.</p>
        <p>Optimized website performance and user experience.</p>
```

```
        <h3>Skills</h3>
        <ul>
            <li>HTML</li>
            <li>CSS</li>
            <li>JavaScript</li>
            <li>PHP</li>
            <li>MySQL</li>
        </ul>
</body>
</html>
```

Create an HTML webpage that shows Poster Presentation using all Table Properties?

```
<!DOCTYPE html>
<html>
<head>
    <title>Poster Presentation</title>
    <style>
        table, th, td {  border: 1px solid
            black;  border-collapse:
            collapse;  padding: 10px;
            text-align: center;
        }
        th {
            background-color: lightgray;  font-
            weight: bold;
        }
        tr:nth-child(even) {  background-
            color: lightblue;
        }
        tr:hover {  background-color:
            yellow;
        }
    </style>
</head>
<body>
    <h1>Poster Presentation</h1>
    <table>
        <thead>
            <tr>
                <th>Presenter Name</th>
                <th>Poster Title</th>
```

```
                    <th>Abstract</th>
                    <th>Keywords</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>John Doe</td>
                    <td>Effects of Climate Change on Arctic Wildlife</td>
                    <td>Climate change is affecting wildlife populations in the Arctic,
with impacts on species such as polar bears, arctic foxes, and reindeer.</td>
                    <td>climate change, Arctic, wildlife, polar bears, arctic foxes,
reindeer</td>
                </tr>
                <tr>
                    <td>Jane Smith</td>
                    <td>The Role of Microbes in Soil Health</td>
                    <td>Microbes play an important role in soil health, influencing
nutrient cycling, plant growth, and carbon sequestration.</td>
                    <td>microbes, soil health, nutrient cycling, plant growth, carbon
sequestration</td>
                </tr>
                <tr>
                    <td>Bob Brown</td>
                    <td>Developing Sustainable Agriculture Practices</td>
                    <td>Sustainable agriculture practices can help reduce
environmental impacts and improve food security.</td>
                    <td>sustainable agriculture, food security, environmental
impacts</td>
                </tr>
            </tbody>
        </table>
</body>
</html>
```

Create an HTML page table and form?

```
<!DOCTYPE html>
<html>
<head>
    <title>Table and Form Example</title>
</head>
<body>
```

```
<h1>Table and Form Example</h1>
<table>
        <thead>
            <tr>
                <th>Name</th>
                <th>Age</th>
                <th>Email</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>John Doe</td>
                <td>30</td>
                <td>john.doe@example.com</td>
            </tr>
            <tr>
                <td>Jane Smith</td>
                <td>25</td>
                <td>jane.smith@example.com</td>
            </tr>
            <tr>
                <td>Bob Brown</td>
                <td>40</td>
                <td>bob.brown@example.com</td>
            </tr>
        </tbody>  </table>
<br>
<form>
        <label for="name">Name:</label>
        <input type="text" id="name" name="name"><br>
        <label for="age">Age:</label>
        <input type="number" id="age" name="age"><br>
        <label for="email">Email:</label>
        <input type="email" id="email" name="email"><br>
        <input type="submit" value="Submit">
</form>
</body>
</html>
```

Create Registration form and do proper validation with HTML 5 inbuilt functionality. (Don't use JavaScript).

```
<!DOCTYPE html>
<html>
<head>
	<title>Registration Form</title>
</head>
<body>
	<h1>Registration Form</h1>
	<form method="post">
		<label for="username">Username:</label>
		<input type="text" id="username" name="username" required minlength="6"
maxlength="20" pattern="[A-Za-z0-9]+"><br>
		<!-- the 'required' attribute ensures that the field is not left empty -->
		<!-- the 'minlength' and 'maxlength' attributes set the minimum and maximum
length of the input -->
		<!-- the 'pattern' attribute specifies a regular expression that the input must
		match
-->

		<label for="email">Email:</label>
		<input type="email" id="email" name="email" required><br>

		<label for="password">Password:</label>
		<input type="password" id="password" name="password" required
minlength="8"><br>

		<label for="confirm_password">Confirm Password:</label>
		<input type="password" id="confirm_password" name="confirm_password"
required minlength="8" onchange="validatePassword()"><br>
		<!-- the 'onchange' attribute specifies a JavaScript function to be called when
		the
value of the field changes -->

		<input type="submit" value="Register">
	</form>

	<script>  function validatePassword() {  if
		(document.getElementById("password").value !=
document.getElementById("confirm_password").value) {

document.getElementById("confirm_password").setCustomValidity("Passwords do not match");
			} else {

document.getElementById("confirm_password").setCustomValidity("");
			}
```

```
            }
        </script>
</body>
</html>
```

Make a Resume using the HTML tags with CSS.?

```
<!DOCTYPE html>
<html>
<head>
        <title>John Doe's Resume</title>
        <style>  body {  font-family: Arial, sans-
                serif;  margin: 0;  padding: 0;
                        background-color: #f5f5f5;
                }
                header {  background-color:
                        #333;
                        color: #fff;
                        padding: 20px;  text-
                        align: center;  font-size:
                        28px;
                }
                .container {
                        max-width: 800px;
                        margin: 0 auto;
                        padding: 20px;
                        background-color: #fff;
                        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
                }
                h1, h2 {  margin:
                        0;
                }
                h1 {  font-size: 36px;
                        color: #333;
                        margin-top:
                        30px;
                }
                h2 {  font-size: 24px;
                        color: #666;
                        margin-top:
                        20px;
                }
                p {
```

```
                margin: 10px 0;  line-
                height: 1.5;
        }
        .skills {
                margin-top: 20px;
        }
        .skills h3 {  margin: 0;
                font-size:
                20px;  color:
                #333;
        }
        .skills ul {
                margin: 10px 0;
                padding: 0;  list-
                style: none;
        }
        .skills li {
                margin: 5px 0;
                padding: 5px;
                background-color:
                #eee;  border-radius:
                5px;
        }
    </style>
</head>
<body>
    <header>
        <h1>John Doe</h1>
        <p>Web Developer</p>
    </header>
    <div class="container">
        <h2>Summary</h2>
        <p>I am an experienced web developer with a passion for creating clean,
 elegant, and efficient code. I specialize in HTML, CSS, JavaScript, and PHP, and I am always
looking for new challenges and opportunities to learn and grow.</p>

        <h2>Skills</h2>
        <div class="skills">
            <h3>Web Development</h3>
            <ul>
                <li>HTML</li>
                <li>CSS</li>
                <li>JavaScript</li>
                <li>PHP</li>
```

```
                </ul>

                <h3>Frameworks & Libraries</h3>
                <ul>
                        <li>Bootstrap</li>
                        <li>jQuery</li>
                        <li>React</li>
                        <li>Vue.js</li>
                </ul>

                <h3>Tools & Technologies</h3>
                <ul>
                        <li>Git</li>
                        <li>Webpack</li>
                        <li>Gulp</li>
                        <li>Sass</li>
                </ul>
        </div>

        <h2>Education</h2>
        <p>Bachelor of Science in Computer Science<br>
        XYZ University, 2010-2014</p>

        <h2>Experience</h2>
        <h3>Web Developer</h3>
        <p>ABC Company, 2015-present</p>
        <ul>
                <li>Developed and maintained company website using HTML, CSS,
JavaScript, and
```

Create an HTML Page containing the following Gray Layout using CSS.??

```
<!DOCTYPE html>
<html>
<head>
        <title>Gray Layout Example</title>
        <style>  body {  background-color:
                #f2f2f2;  margin: 0;  padding: 0;
                }

                .container {  max-width: 960px;  margin: 0
                        auto;  padding: 20px;  background-
```

```
                color: #fff;  box-shadow: 0 0 10px
                rgba(0,0,0,0.2);
            }

            h1 {  font-size: 36px;
                font-weight:
                bold;  color:
                #333;  margin-
                top: 0;
            }

            p {
                 font-size: 18px;  line-
                height: 1.5;  color:
                #666;
            }

            .btn {  display: inline-block;
                padding: 10px 20px;
                background-color:
                #333;
                 color: #fff;
                 text-decoration: none;  border-
                radius: 5px;
                 transition: all 0.3s ease-in-out;
            }

            .btn:hover {  background-color:
                #666;
                 color: #fff;
            }
        </style>
</head>
<body>
        <div class="container">
            <h1>Welcome to our website!</h1>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed ac erat ut nunc
fringilla accumsan. Morbi egestas quam id velit molestie, non vestibulum leo
        dictum.</p>  <a href="#" class="btn">Learn more</a>  </div>
</body>
</html>
```

Demonstrate JavaScript Form Validation with proper examples.?

```
// Get the form element  const form =
document.getElementById("myForm");

// Get the input fields  const nameInput =
document.getElementById("name");  const emailInput =
document.getElementById("email");  const phoneInput =
document.getElementById("phone");  const passwordInput =
document.getElementById("password");

// Add an event listener for form submission
form.addEventListener("submit", (event) => {
// Prevent the form from submitting
event.preventDefault();

  // Validate the name field  if
 (nameInput.value.trim() === "") {
    alert("Name field is required.");
   return;
  }

  // Validate the email field
  if (!validateEmail(emailInput.value)) {
    alert("Email is not valid.");
   return;
  }

  // Validate the phone field  if
 (!validatePhone(phoneInput.value)) {
    alert("Phone number is not valid.");
   return;
  }

  // Validate the password field  if
 (passwordInput.value.trim() === "") {
    alert("Password field is required.");
   return;
  }

  // Submit the form if all fields are valid
 alert("Form submitted successfully!");
 form.submit();
});
```

```
// Function to validate email
function validateEmail(email) {
   const regex = /\\S+@\\S+\\.\\S+/;
  return regex.test(email);
}

// Function to validate phone number
function validatePhone(phone) {
   const regex = /^[0-9]{10}$/;
  return regex.test(phone);
}
```

Write a javascript to check if the number is even or odd.?

```
<!DOCTYPE html>
<html>
  <head>
    <title>Check Even or Odd</title>
    <script>  function
     checkNumber() {
        // Get the value of the input field  var num =
       document.getElementById("num").value;  // Check
       if the number is even or odd  if (num % 2 == 0) {
          alert(num + " is even.");
       } else {  alert(num + "
        is odd.");
       }
     }
    </script>
  </head>
  <body>
    <h1>Check Even or Odd</h1>
    <form>
      <label for="num">Enter a number:</label>
      <input type="number" id="num" name="num" required><br><br>
      <button type="button" onclick="checkNumber()">Check</button>
   </form>
  </body>
</html>
```

Create a page and access the LocationAPI.?

```
<!DOCTYPE html>
```

```
<html>
  <head>
    <title>Location API Example</title>
    <script>  function
     getLocation() {
        // Check if the browser supports
       geolocation  if (navigator.geolocation) {  //
       Get the current position of the user
         navigator.geolocation.getCurrentPosition(showPosition);
       } else {  alert("Geolocation is not supported by this
        browser.");
       }
      }

      function showPosition(position) {
       // Get the latitude and longitude of the user's
       position  var lat = position.coords.latitude;  var lon =
       position.coords.longitude;

       // Display the latitude and longitude in an HTML element
       var locationDiv = document.getElementById("location");
       locationDiv.innerHTML =
          "Latitude: " + lat + "<br>Longitude: " + lon;
      }
    </script>
  </head>
  <body>
    <h1>Location API Example</h1>
    <button type="button" onclick="getLocation()">Get Location</button>
   <div id="location"></div>
  </body>
</html>
```

Create a simple XMLHTTPRequest,and retrieve the data from the text file.?

```
<!DOCTYPE html>
<html>
  <head>
    <title>XMLHttpRequest Example</title>
    <script>  function
     loadData() {
        // Create a new XMLHttpRequest object
       var xhttp = new XMLHttpRequest();
```

```
        // Set the onreadystatechange function to handle the response
        xhttp.onreadystatechange = function() {
          if (this.readyState == 4 && this.status == 200) {  // Display the
           response text in an HTML element
           document.getElementById("data").innerHTML = this.responseText;
          }
        };

        // Open a GET request to the text file
        xhttp.open("GET", "data.txt", true);

        // Send the request
        xhttp.send();
      }
    </script>
  </head>
  <body>
    <h1>XMLHttpRequest Example</h1>
    <button type="button" onclick="loadData()">Load Data</button>
   <div id="data"></div>
  </body>
</html>
```

To study DDL-create and DML-insert commands.?

DDL and DML are two types of SQL commands. DDL stands for Data Definition Language, and it is used to create and modify the structure of database objects, such as tables, indexes, and views. DML stands for Data Manipulation Language, and it is used to insert, update, and delete data in a database.

Here are some examples of DDL and DML commands:

DDL - CREATE TABLE:

The CREATE TABLE statement is used to create a new table in a database. Here is an example:

SQLCopy code

```
CREATE TABLE customers (
  id INT PRIMARY KEY,
  name VARCHAR(50),
  email VARCHAR(50),
  phone VARCHAR(20)
);
```

This statement creates a new table named "customers" with four columns: id, name, email, and phone. The id column is defined as the primary key, which means that it will contain a unique value for each row in the table.

DDL - ALTER TABLE:

The ALTER TABLE statement is used to modify the structure of an existing table in a database. Here is an example:

SQLCopy code
```
ALTER TABLE customers
ADD address VARCHAR(100);
```

This statement adds a new column named "address" to the "customers" table.

DML - INSERT INTO:
The INSERT INTO statement is used to insert new rows into a table. Here is an example:

SQLCopy code
```
INSERT INTO customers (id, name, email, phone)
VALUES (1, 'John Doe', 'john.doe@example.com', '555-1234');
```

This statement inserts a new row into the "customers" table with the specified values for the id, name, email, and phone columns.

DML - UPDATE:

The UPDATE statement is used to modify existing rows in a table. Here is an example:

SQLCopy code
```
UPDATE customers
SET phone = '555-5678'
WHERE id = 1;
```

This statement updates the "phone" column for the row with id 1 in the "customers" table.

DML - DELETE:

The DELETE statement is used to delete rows from a table. Here is an example:

SQLCopy code
```
DELETE FROM customers
WHERE id = 1;
```

This statement deletes the row with id 1 from the "customers" table.

Create tables and insert sample data in tables.?

DDL - CREATE TABLE:

SQLCopy code
```
CREATE TABLE employees (
   id INT PRIMARY KEY,
   name VARCHAR(50),
    age INT,  department
   VARCHAR(50),  salary
   DECIMAL(10,2)
);

CREATE TABLE departments
   (  id INT PRIMARY KEY,
   name VARCHAR(50),
   location VARCHAR(50)
);
```

This statement creates two tables: "employees" and "departments". The "employees" table has five columns: id, name, age, department, and salary. The "departments" table has three columns: id, name, and location.

DML - INSERT INTO:

SQLCopy code
```
INSERT INTO employees (id, name, age, department, salary)
VALUES (1, 'John Doe', 30, 'IT', 5000.00);

INSERT INTO employees (id, name, age, department, salary)
VALUES (2, 'Jane Smith', 25, 'HR', 4000.00);

INSERT INTO employees (id, name, age, department, salary)
VALUES (3, 'Bob Johnson', 40, 'Finance', 6000.00);
```

```
INSERT INTO departments (id, name, location)
VALUES (1, 'IT', 'New York');

INSERT INTO departments (id, name, location)
VALUES (2, 'HR', 'Chicago');

INSERT INTO departments (id, name, location)
VALUES (3, 'Finance', 'Los Angeles');
```

These statements insert sample data into the "employees" and "departments" tables. The first three INSERT statements add three employees to the "employees" table, and the last three INSERT statements add three departments to the "departments" table.

You can query these tables to view the data using SELECT statements:

```
SQLCopy code
SELECT * FROM employees;

SELECT * FROM departments;
```

These statements will display all the rows in the "employees" and "departments" tables, respectively.

Write the SQL queries to provide constraints on given tables.?

1. Adding a primary key constraint to a table:

```
SQLCopy code
ALTER TABLE employees
ADD CONSTRAINT pk_employees_id PRIMARY KEY (id);
```

This statement adds a primary key constraint to the "employees" table, using the "id" column as the primary key.

1. Adding a foreign key constraint to a table:

```
SQLCopy code
ALTER TABLE employees
ADD CONSTRAINT fk_employees_department
```

```
FOREIGN KEY (department_id)
REFERENCES departments(id);
```

This statement adds a foreign key constraint to the "employees" table, using the "department_id" column as the foreign key. The foreign key references the "id" column of the "departments" table.

1. Adding a unique constraint to a table:

```
SQLCopy code
ALTER TABLE employees
ADD CONSTRAINT uc_employees_email UNIQUE (email);
```

This statement adds a unique constraint to the "employees" table, using the "email" column as the unique key.

1. Adding a check constraint to a table:

```
SQLCopy code
ALTER TABLE employees
ADD CONSTRAINT ck_employees_salary CHECK (salary > 0);
```

This statement adds a check constraint to the "employees" table, ensuring that the "salary" column is greater than zero.

1. Adding a not null constraint to a table:

```
SQLCopy code
ALTER TABLE employees
ALTER COLUMN name SET NOT NULL;
```

This statement adds a not null constraint to the "name" column of the "employees" table. This means that a value must be provided for the "name" column when inserting or updating rows.

Write the SQL queries to perform various aggregate functions on table data?

Finding the sum of a column:
SQL

Copy code
SELECT SUM(salary) as total_salary
FROM employees;
This statement finds the sum of the "salary" column in the "employees" table and displays the result as "total_salary".

Finding the average of a column:
SQL
Copy code
SELECT AVG(age) as avg_age
FROM employees;
This statement finds the average of the "age" column in the "employees" table and displays the result as "avg_age".

Finding the minimum value in a column:
SQL
Copy code
SELECT MIN(salary) as min_salary
FROM employees;
This statement finds the minimum value in the "salary" column in the "employees" table and displays the result as "min_salary".

Finding the maximum value in a column:
SQL
Copy code
SELECT MAX(salary) as max_salary
FROM employees;
This statement finds the maximum value in the "salary" column in the "employees" table and displays the result as "max_salary".

Counting the number of rows in a table:
SQL
Copy code
SELECT COUNT(*) as total_rows
FROM employees;
This statement counts the number of rows in the "employees" table and displays the result as "total_rows". Note that we use the "*" wildcard to count all rows in the table.

Write the SQL queries to perform numeric,date and String functions.?

1. Numeric functions:

SQLCopy code

```
SELECT ABS(-10) as absolute_value; -- Returns 10 (absolute value)
SELECT CEILING(3.14) as ceiling_value; -- Returns 4 (next highest integer)
SELECT FLOOR(3.99) as floor_value; -- Returns 3 (next lowest integer)
SELECT ROUND(3.75) as rounded_value; -- Returns 4 (rounded to nearest integer)
SELECT POWER(2, 3) as power_value; -- Returns 8 (2 raised to the power of 3)
```

1. Date functions:

```
SQLCopy code
SELECT NOW() as current_time; -- Returns the current date and time
SELECT YEAR('2023-05-20') as year_value; -- Returns 2023 (year from the date)
SELECT MONTH('2023-05-20') as month_value; -- Returns 5 (month from the date)
SELECT DAY('2023-05-20') as day_value; -- Returns 20 (day from the date)
SELECT DATEDIFF('2023-05-20', '2023-05-01') as date_diff; -- Returns 19 (difference between two dates)
```

1. String functions:

```
SQLCopy code
SELECT CONCAT('Hello', ' ', 'World') as concat_string; -- Returns 'Hello World' (concatenation of two strings)
SELECT SUBSTRING('Hello World', 7, 5) as substring_value; -- Returns 'World' (substring of a string)
SELECT UPPER('hello world') as upper_string; -- Returns 'HELLO WORLD' (converts to uppercase)
SELECT LOWER('HELLO WORLD') as lower_string; -- Returns 'hello world' (converts to lowercase)
SELECT LENGTH('Hello World') as length_value; -- Returns 11 (length of a string)
```

Note that these examples may not be supported in all SQL databases, as the syntax may vary depending on the database being used.