INDIRA GANDHI NATIONAL OPEN UNIVERSITY

ignou THE PEOPLE'S UNIVERSITY

LABORATORY RECORD

Month &Year : ………………………………………………….….…

Name : ……………………………………………………………….

Study Center : 1402, SH College, Thevara, Kochi-13

Course Code : ……………..……………………………………………

Course Title : ……………………………………………...……..….

…………………………………………………………..…………

Program Code: ……………………..……………………………………

Enrolment No: …………………………………………………………

External Examiner Staff In-Charge

INDEX

Sl No.	Description	Page No.

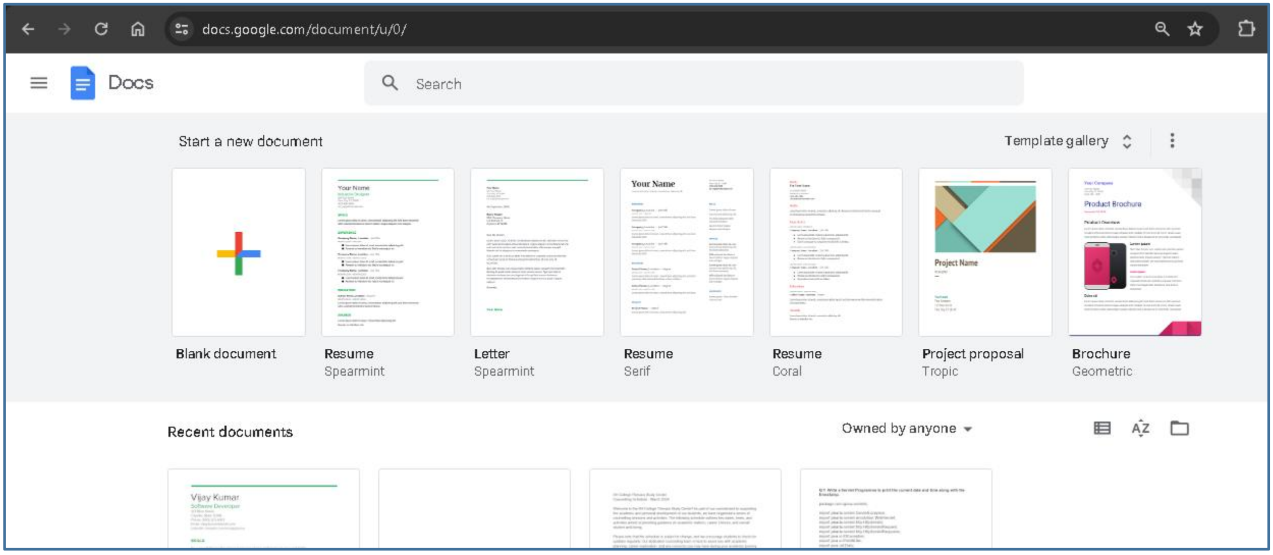
Section 1: Cloud Computing Lab

Question 1:

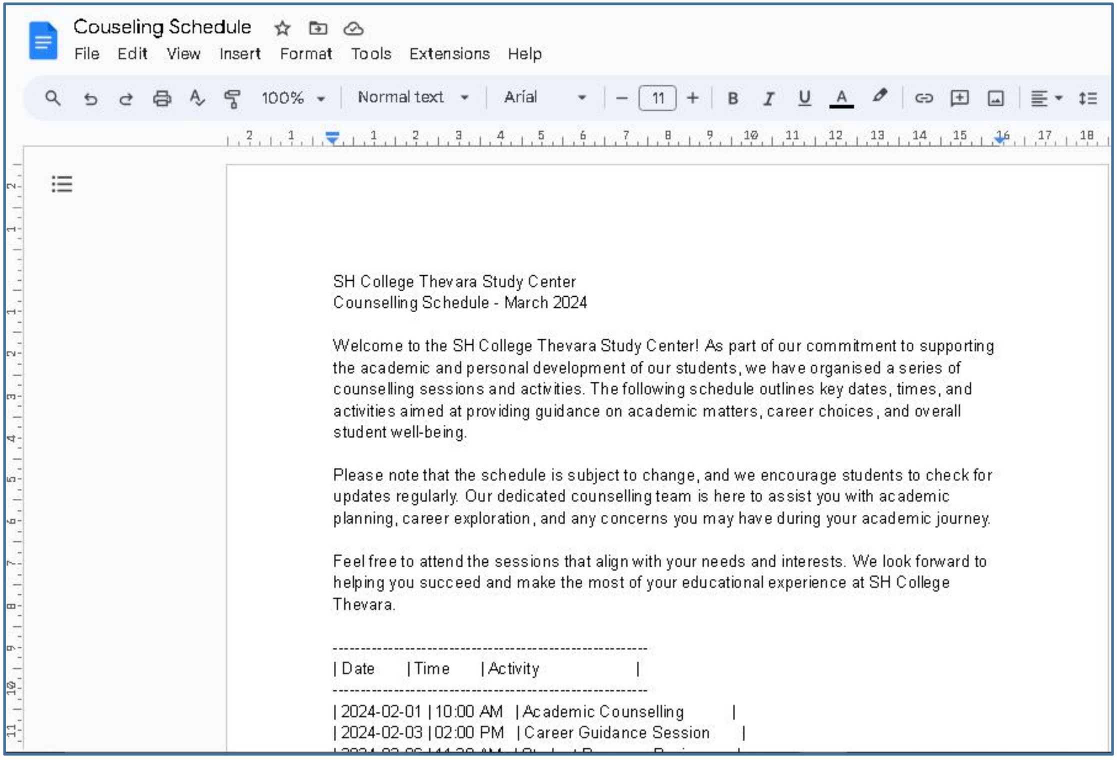
Create a word document of your counselling schedule of the study centre and store locally and on Google Drive with doc and pdf formats. Share it with your peer and faculty in View mode.

Answer:

# Google Docs Steps:
1. Log in with your Google account.
2. Go to Google Docs: https://docs.google.com



# Create and Save Document:
3. Click on "Blank" to create a new document.
4. Enter counseling schedule details.



SH College Thevara Study Center
Counselling Schedule - March 2024

Welcome to the SH College Thevara Study Center! As part of our commitment to supporting the academic and personal development of our students, we have organised a series of counselling sessions and activities. The following schedule outlines key dates, times, and activities aimed at providing guidance on academic matters, career choices, and overall student well-being.

Please note that the schedule is subject to change, and we encourage students to check for updates regularly. Our dedicated counselling team is here to assist you with academic planning, career exploration, and any concerns you may have during your academic journey.

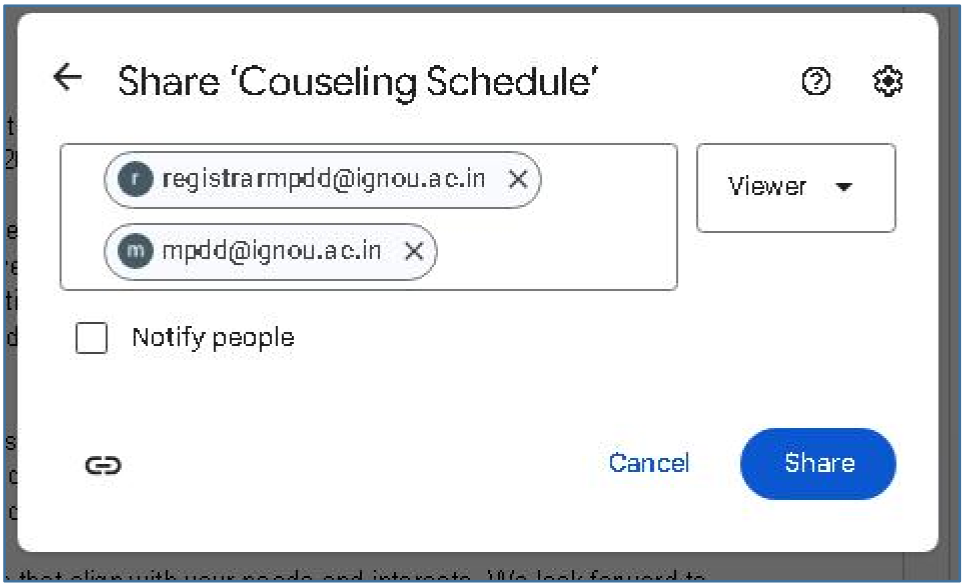
Feel free to attend the sessions that align with your needs and interests. We look forward to helping you succeed and make the most of your educational experience at SH College Thevara.

```
--------------------------------------------------------
| Date       | Time       | Activity                   |
--------------------------------------------------------
| 2024-02-01 | 10:00 AM  | Academic Counselling        |
| 2024-02-03 | 02:00 PM  | Career Guidance Session     |
```

5. Save the document to Google Drive.

Share Google Docs Document:
6. Click "Share" in the upper-right corner.

7. Enter peer and faculty email addresses.
8. Set permissions to "Viewer."
9. Click "Send."



Share 'Couseling Schedule'

registrarmpdd@ignou.ac.in

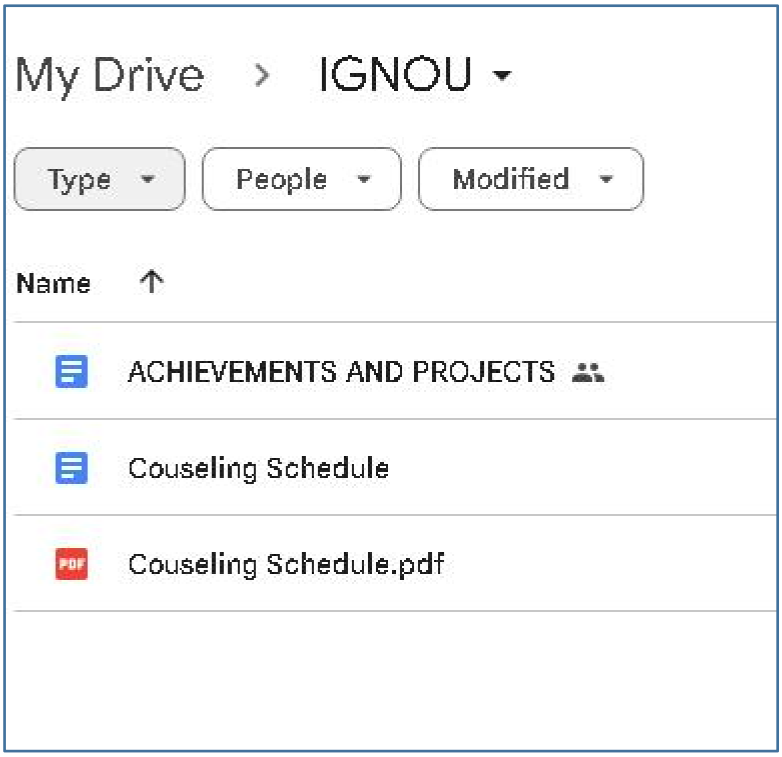
mpdd@ignou.ac.in

Viewer

□ Notify people

# Export to PDF:

10. While in the document, click on "File" -> "Download" -> "PDF Document (.pdf)".



My Drive > IGNOU

ACHIEVEMENTS AND PROJECTS

Couseling Schedule

Couseling Schedule.pdf

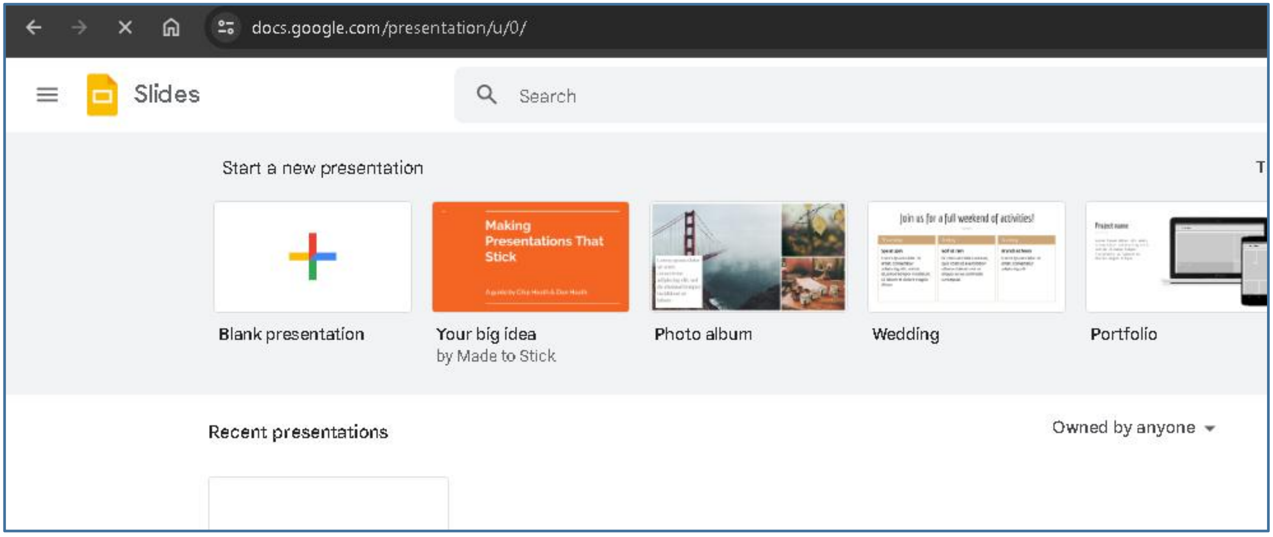
Question 2:

Using Google Slides, prepare a Presentation consisting of at least 20 slides on Cloud Computing covering introduction, models, services, architecture, applications and security aspects.

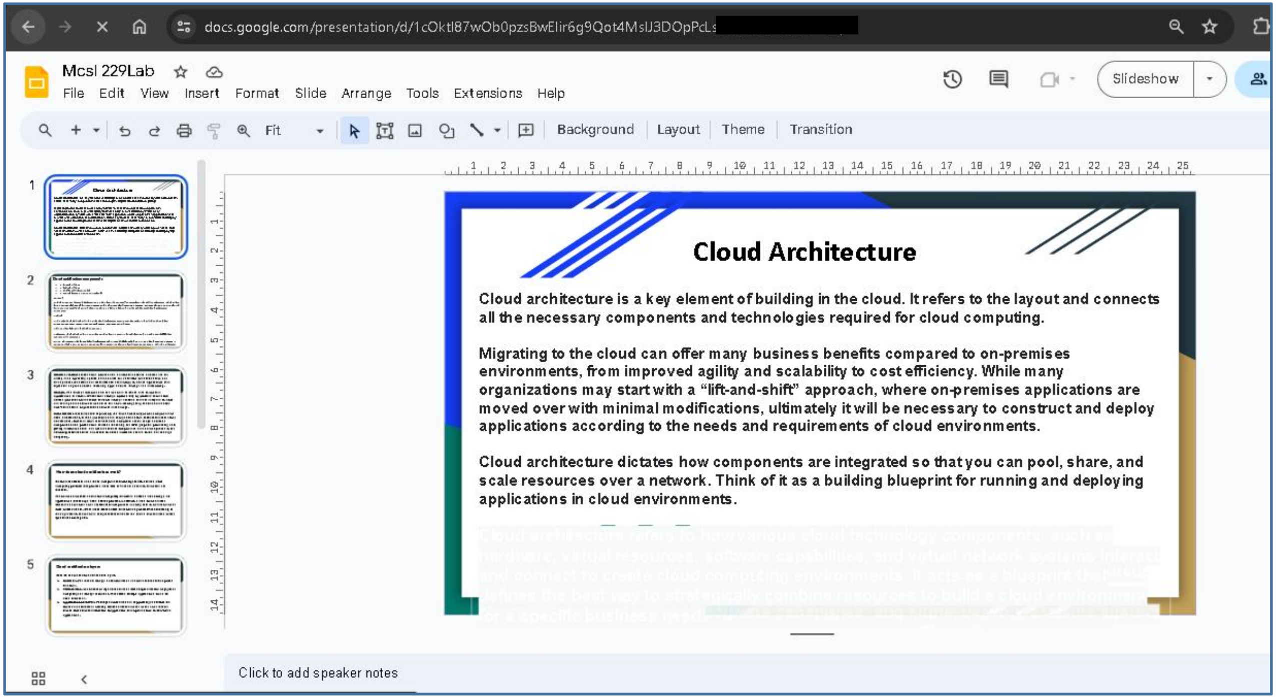
Answer:

1. Log in to Google Slides with your account.

2. Start a new presentation, selecting a blank template.



3. Develop a comprehensive presentation on Cloud Computing covering introduction, models, services, architecture, applications, security aspects, and a conclusion with a Q&A session.


docs.google.com/presentation/d/1cOktl87wOb0pzsBwElir6g9Qot4MslJ3DOpPcLs[redacted]
Mcsl 229Lab
File Edit View Insert Format Slide Arrange Tools Extensions Help
Slideshow
Fit
Background Layout Theme Transition
Click to add speaker notes

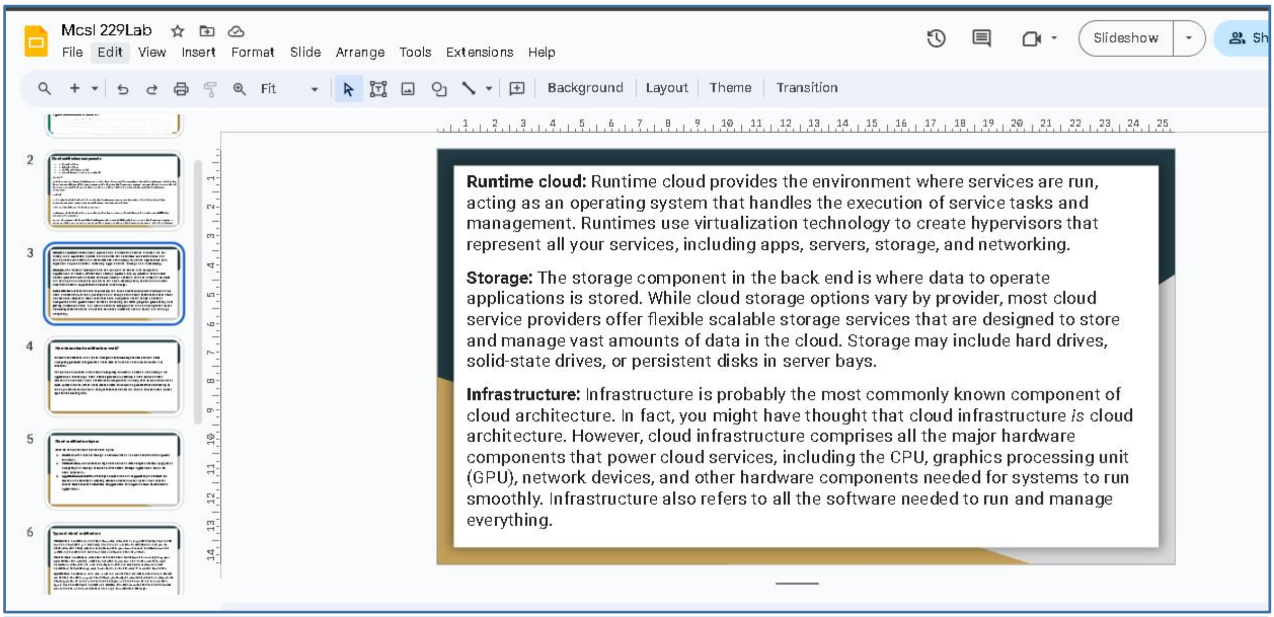
Cloud Architecture

Cloud architecture is a key element of building in the cloud. It refers to the layout and connects all the necessary components and technologies required for cloud computing.

Migrating to the cloud can offer many business benefits compared to on-premises environments, from improved agility and scalability to cost efficiency. While many organizations may start with a "lift-and-shift" approach, where on-premises applications are moved over with minimal modifications, ultimately it will be necessary to construct and deploy applications according to the needs and requirements of cloud environments.

Cloud architecture dictates how components are integrated so that you can pool, share, and scale resources over a network. Think of it as a building blueprint for running and deploying applications in cloud environments.

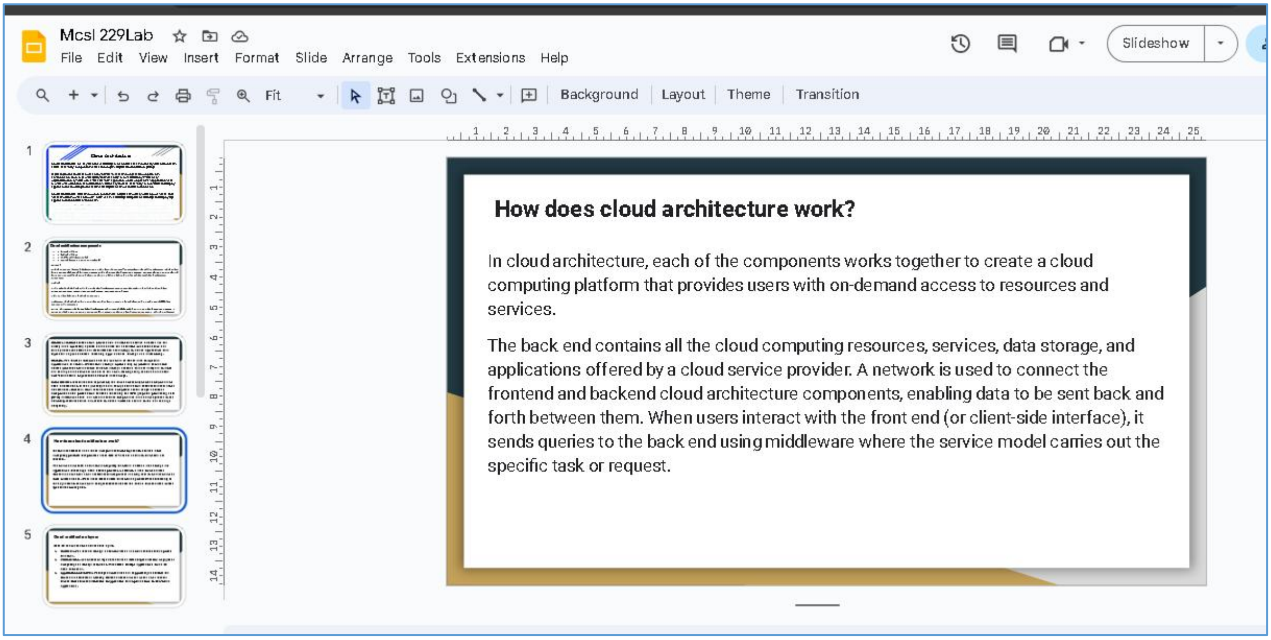
Cloud architecture refers to how various cloud technology components, such as hardware, virtual resources, software capabilities, and virtual network systems interact and connect to create cloud computing environments. It acts as a blueprint that defines the best way to strategically combine resources to build a cloud environment for a specific business need.


Mcsl 229Lab
File Edit View Insert Format Slide Arrange Tools Extensions Help
Slideshow
Fit
Background Layout Theme Transition

Runtime cloud: Runtime cloud provides the environment where services are run, acting as an operating system that handles the execution of service tasks and management. Runtimes use virtualization technology to create hypervisors that represent all your services, including apps, servers, storage, and networking.

Storage: The storage component in the back end is where data to operate applications is stored. While cloud storage options vary by provider, most cloud service providers offer flexible scalable storage services that are designed to store and manage vast amounts of data in the cloud. Storage may include hard drives, solid-state drives, or persistent disks in server bays.

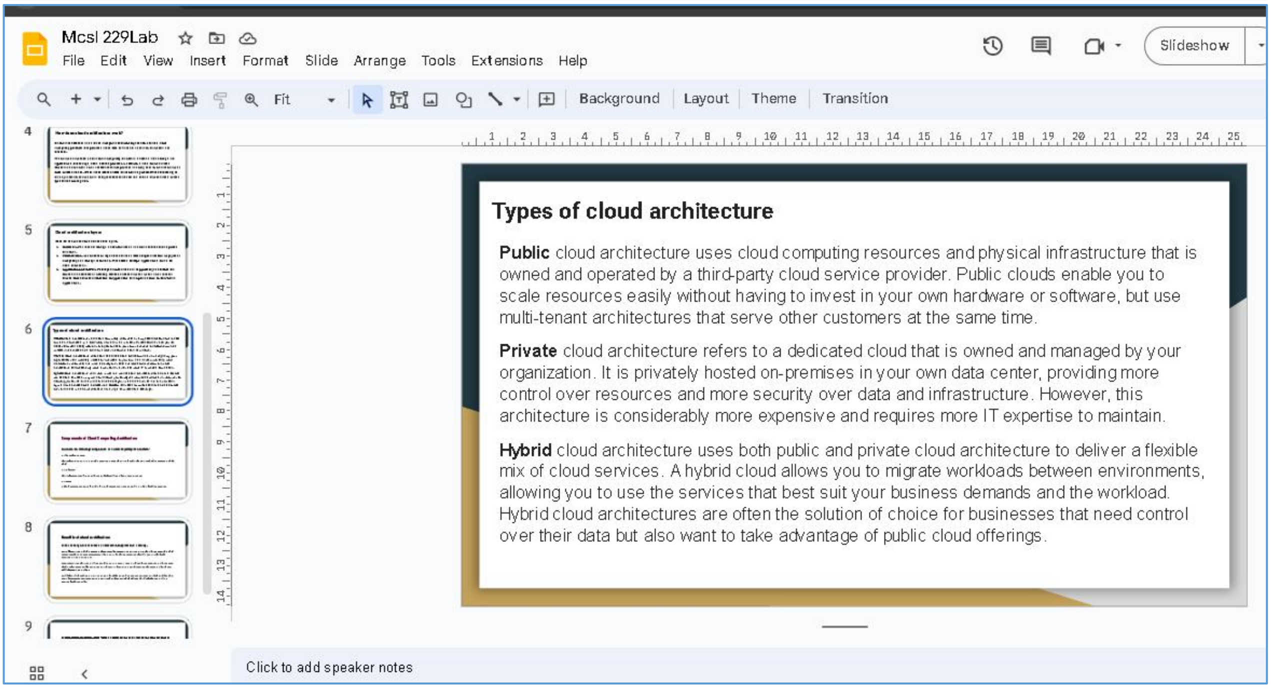
Infrastructure: Infrastructure is probably the most commonly known component of cloud architecture. In fact, you might have thought that cloud infrastructure *is* cloud architecture. However, cloud infrastructure comprises all the major hardware components that power cloud services, including the CPU, graphics processing unit (GPU), network devices, and other hardware components needed for systems to run smoothly. Infrastructure also refers to all the software needed to run and manage everything.


Mcsl 229Lab
File Edit View Insert Format Slide Arrange Tools Extensions Help
Slideshow
Fit
Background Layout Theme Transition

How does cloud architecture work?

In cloud architecture, each of the components works together to create a cloud computing platform that provides users with on-demand access to resources and services.

The back end contains all the cloud computing resources, services, data storage, and applications offered by a cloud service provider. A network is used to connect the frontend and backend cloud architecture components, enabling data to be sent back and forth between them. When users interact with the front end (or client-side interface), it sends queries to the back end using middleware where the service model carries out the specific task or request.


Mcsl 229Lab
File Edit View Insert Format Slide Arrange Tools Extensions Help
Slideshow
Fit
Background Layout Theme Transition
Click to add speaker notes

Types of cloud architecture

Public cloud architecture uses cloud computing resources and physical infrastructure that is owned and operated by a third-party cloud service provider. Public clouds enable you to scale resources easily without having to invest in your own hardware or software, but use multi-tenant architectures that serve other customers at the same time.

Private cloud architecture refers to a dedicated cloud that is owned and managed by your organization. It is privately hosted on-premises in your own data center, providing more control over resources and more security over data and infrastructure. However, this architecture is considerably more expensive and requires more IT expertise to maintain.

Hybrid cloud architecture uses both public and private cloud architecture to deliver a flexible mix of cloud services. A hybrid cloud allows you to migrate workloads between environments, allowing you to use the services that best suit your business demands and the workload. Hybrid cloud architectures are often the solution of choice for businesses that need control over their data but also want to take advantage of public cloud offerings.

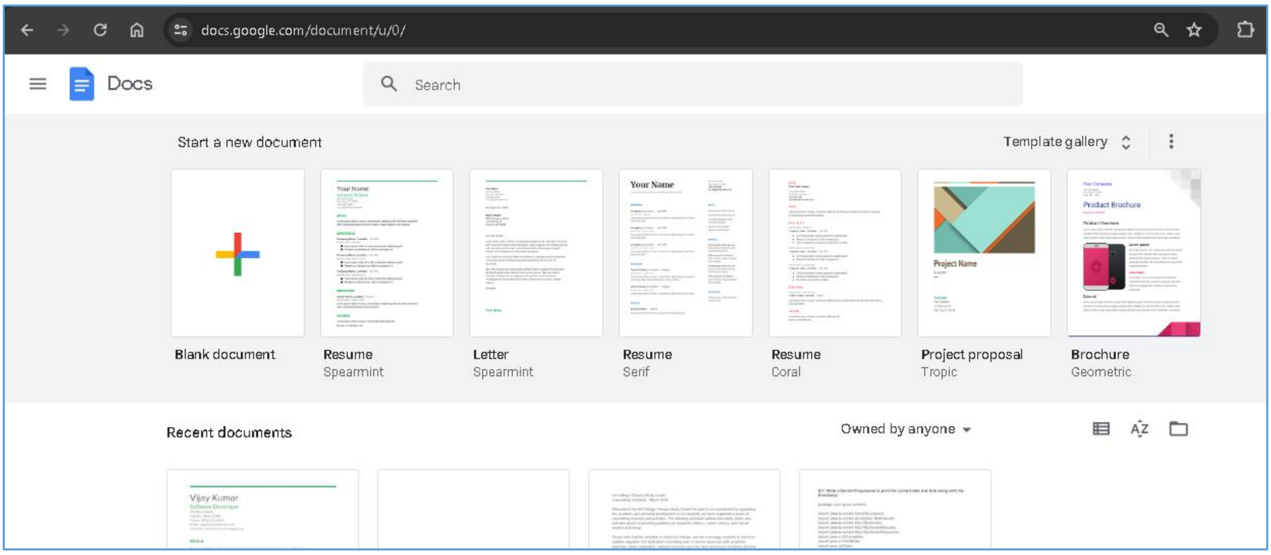
Question 3:

Using Google Docs, create your Resume in a neat format using Google and Zoho cloud.

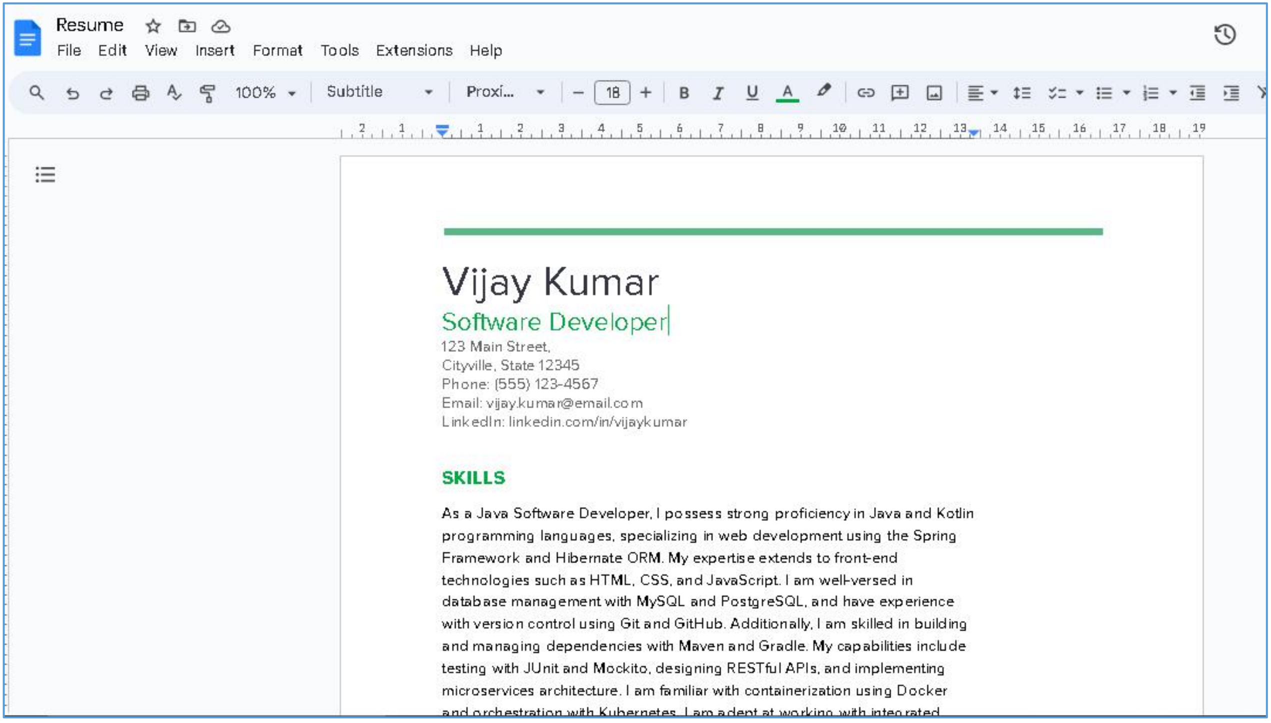
Answer:

Google Docs:

1. Log in to [Google Docs](https://docs.google.com/).
2. Create and format a new document.



3. Input content for Contact Information, Summary, Education, Experience, Skills.



Vijay Kumar

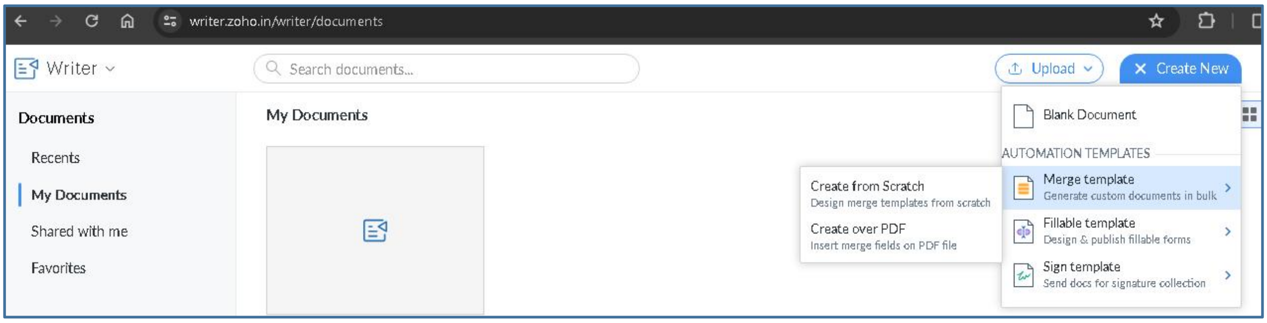
Software Developer

123 Main Street,
Cityville, State 12345
Phone: (555) 123-4567
Email: vijay.kumar@email.com
LinkedIn: linkedin.com/in/vijaykumar

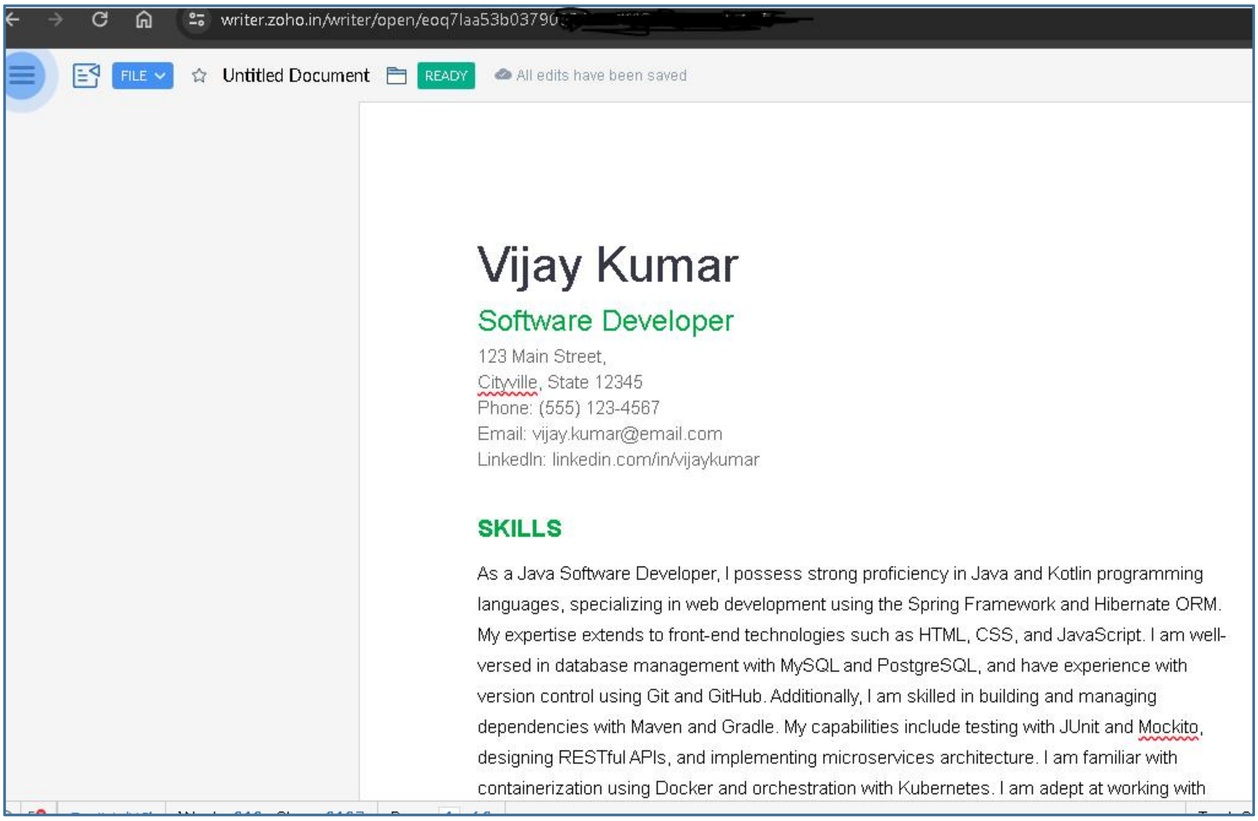
SKILLS

As a Java Software Developer, I possess strong proficiency in Java and Kotlin programming languages, specializing in web development using the Spring Framework and Hibernate ORM. My expertise extends to front-end technologies such as HTML, CSS, and JavaScript. I am well-versed in database management with MySQL and PostgreSQL, and have experience with version control using Git and GitHub. Additionally, I am skilled in building and managing dependencies with Maven and Gradle. My capabilities include testing with JUnit and Mockito, designing RESTful APIs, and implementing microservices architecture. I am familiar with containerization using Docker and orchestration with Kubernetes. I am adept at working with integrated

Zoho Writer:
1. Log in to [Zoho Writer](https://www.zoho.com/writer/).
2. Create and format a new document.



3. Input content for Contact Information, Summary, Education, Experience, Skills.


writer.zoho.in/writer/open/eoq7laa53b03790[redacted]
FILE
Untitled Document
READY
All edits have been saved

Vijay Kumar

Software Developer

123 Main Street,
Cityville, State 12345
Phone: (555) 123-4567
Email: vijay.kumar@email.com
LinkedIn: linkedin.com/in/vijaykumar

SKILLS

As a Java Software Developer, I possess strong proficiency in Java and Kotlin programming languages, specializing in web development using the Spring Framework and Hibernate ORM. My expertise extends to front-end technologies such as HTML, CSS, and JavaScript. I am well-versed in database management with MySQL and PostgreSQL, and have experience with version control using Git and GitHub. Additionally, I am skilled in building and managing dependencies with Maven and Gradle. My capabilities include testing with JUnit and Mockito, designing RESTful APIs, and implementing microservices architecture. I am familiar with containerization using Docker and orchestration with Kubernetes. I am adept at working with

Question 4:

Explore Amazon Drive and NordLocker file storage and sharing solutions. Use only their trail versions.

Answer:

Amazon drive cannot be accessed now, Amazon Photos is available.

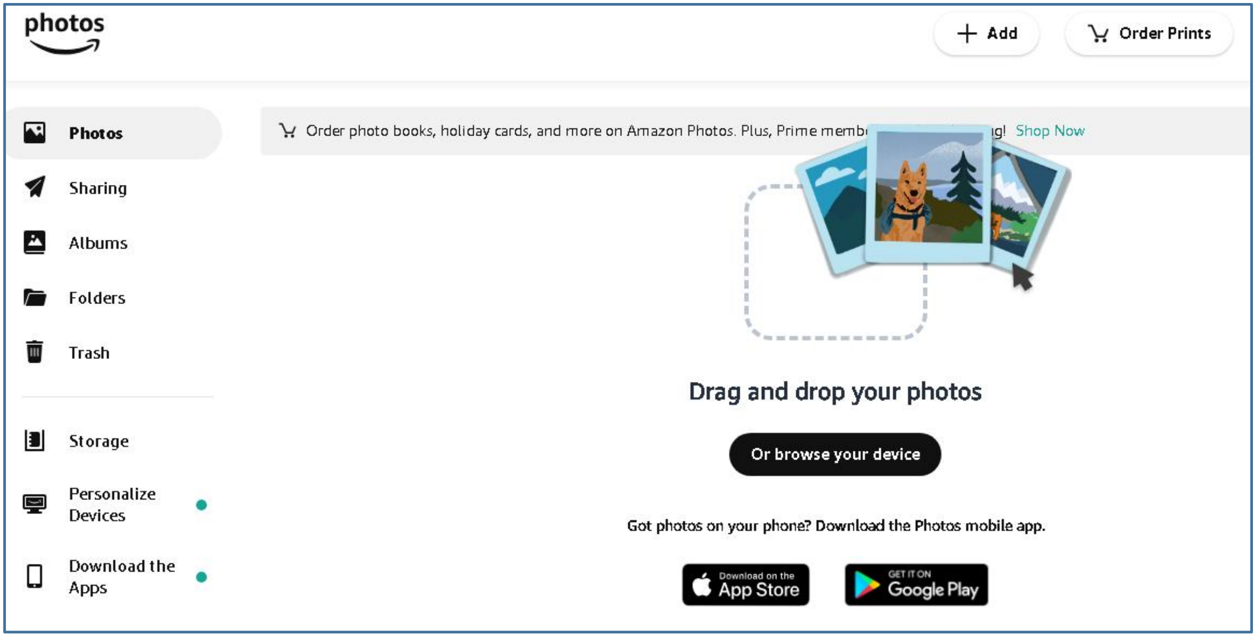
Amazon Photos is a cloud-based photo storage service offered by Amazon, allowing users to securely store and organize their digital images. Users can upload photos and videos from various devices, ensuring a centralized and easily accessible repository. The service provides automatic backup, helping users safeguard their cherished memories. Amazon Photos includes features like facial recognition and object detection, enabling efficient organization and search functionalities. Prime members often enjoy additional benefits, such as unlimited photo storage as part of their subscription.

Accessing Amazon Photos:

1. **Visit Amazon Photos:**
   - Go to [Amazon Photos](https://www.amazon.com/photos) using a web browser.

2. **Log in:**
   - Sign in with the Amazon account associated with your Amazon Photos.

3. **Navigate Your Photos:**
   - Explore your photos and albums using the navigation options on the Amazon Photos website.

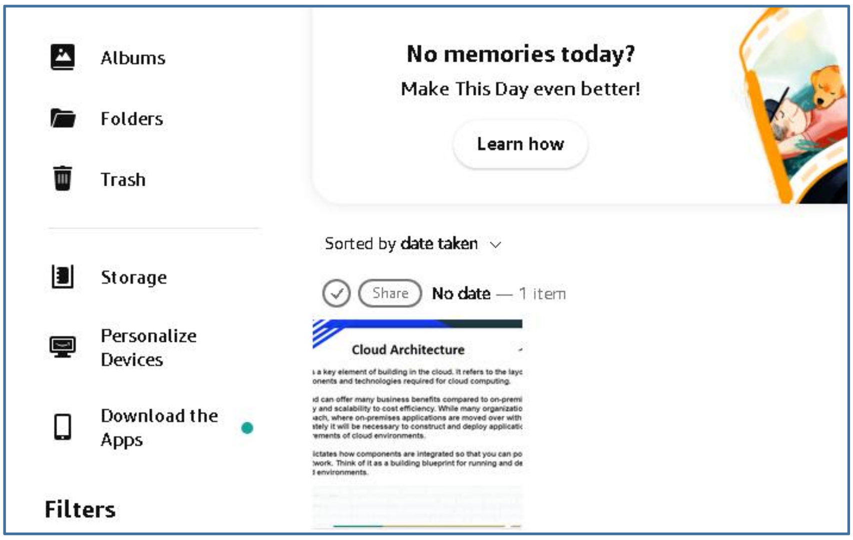


Order photo books, holiday cards, and more on Amazon Photos. Plus, Prime memb g! Shop Now

Drag and drop your photos

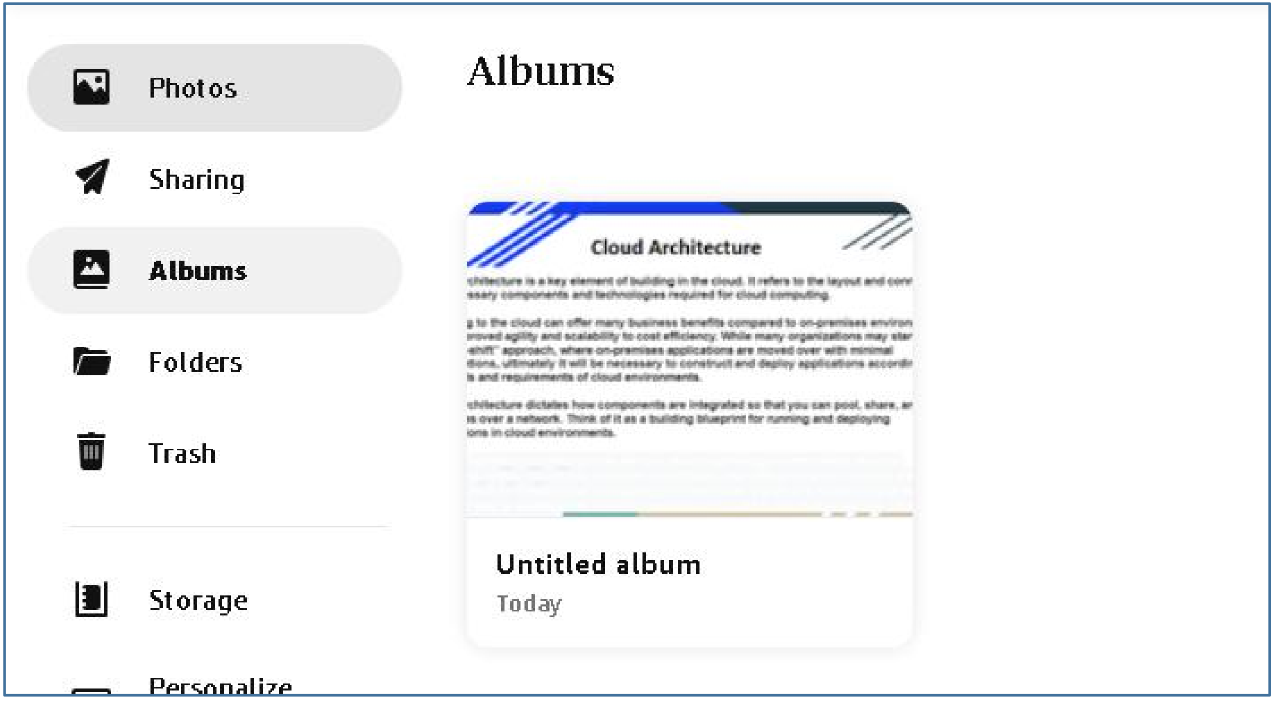
Got photos on your phone? Download the Photos mobile app.

4. **Upload Photos:**
   - Use the "Upload" feature to add new photos to your Amazon Photos library.

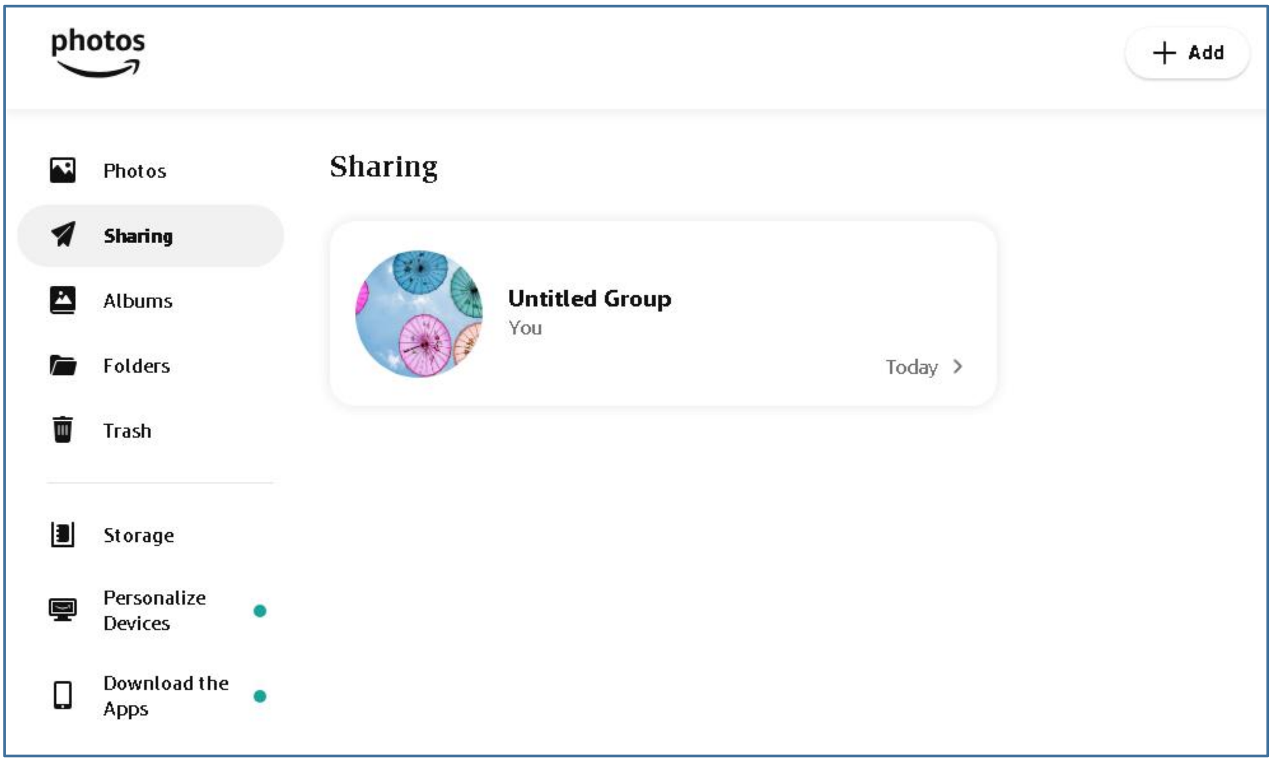


No memories today?
Make This Day even better!

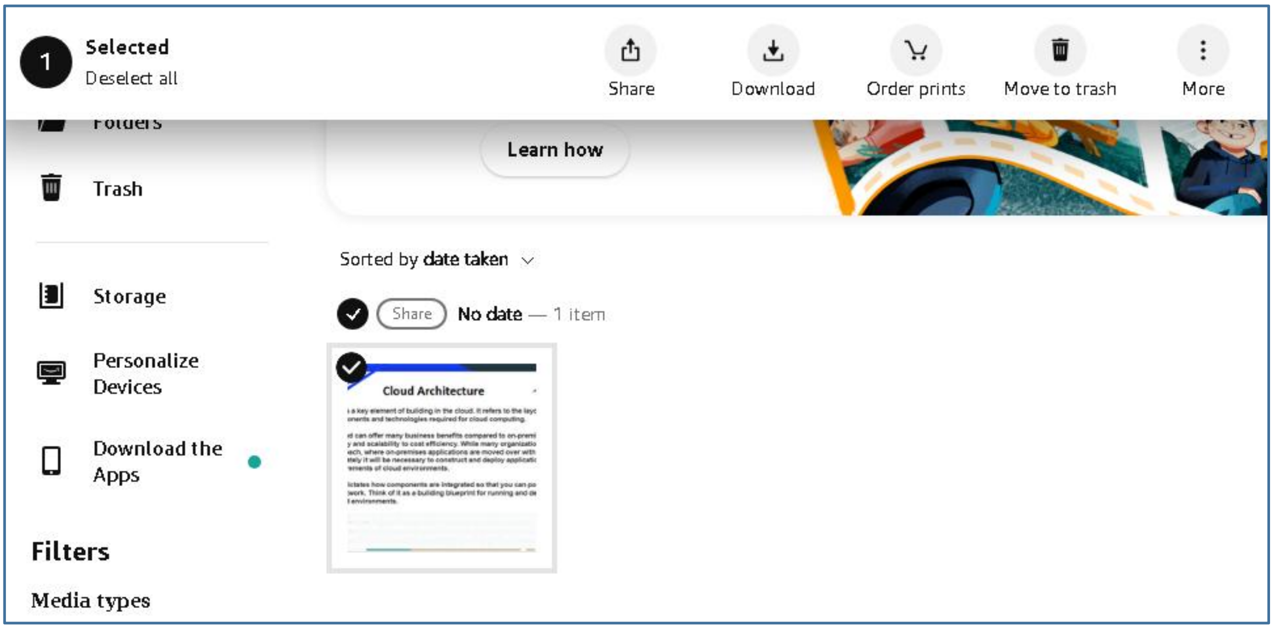
5. **Create Albums:**
   - Organize your photos by creating albums within Amazon Photos.



6. **Share Photos:**
   - Share your photos with others using the sharing options provided in Amazon Photos.



7. **Download Photos:**
   - Download photos to your device if needed, using the download options.

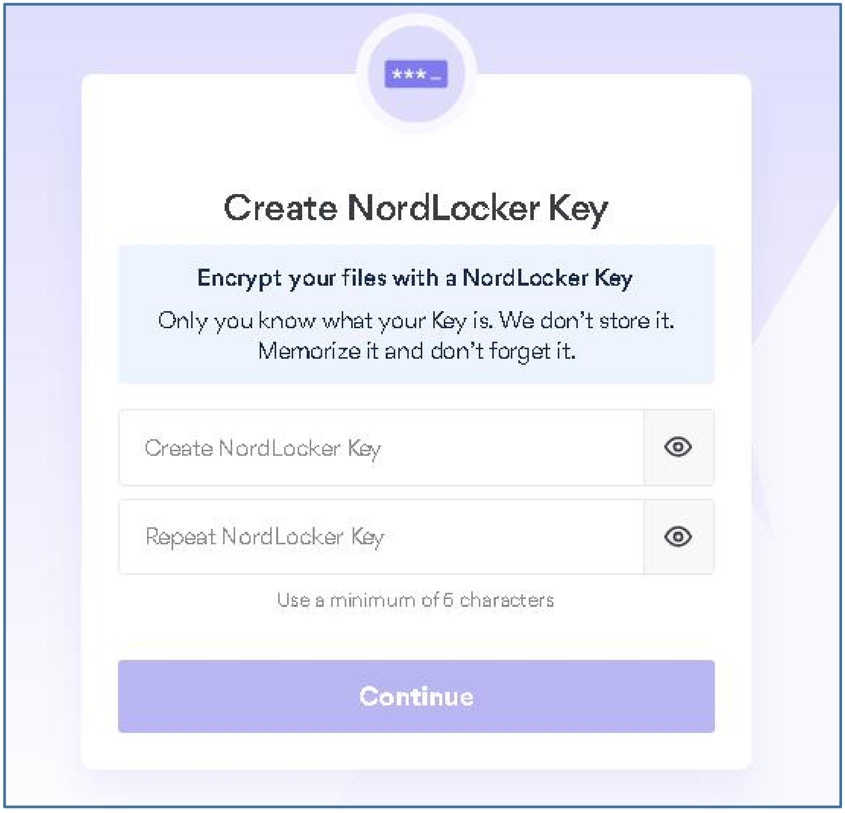


### Accessing NordLocker:

NordLocker is a file encryption and security solution developed by the creators of NordVPN. It is designed to protect users' sensitive files and data with robust encryption algorithms, ensuring privacy and security. NordLocker allows users to create secure folders that are encrypted locally on their devices before being uploaded to the cloud for additional protection. The service supports end-to-end encryption, meaning only the user with the encryption key can access the files. NordLocker is known for its user-friendly interface and seamless integration, providing a convenient and effective solution for users seeking enhanced file security.

Exploring NordLocker involves accessing and managing your encrypted files.
1.. **Create an Account:**
 - Open NordLocker and create a new account.
 - Follow the on-screen instructions to complete the account creation process.

3. **Sign In:**
 - Log in to NordLocker using your newly created account credentials.



Create NordLocker Key

Encrypt your files with a NordLocker Key

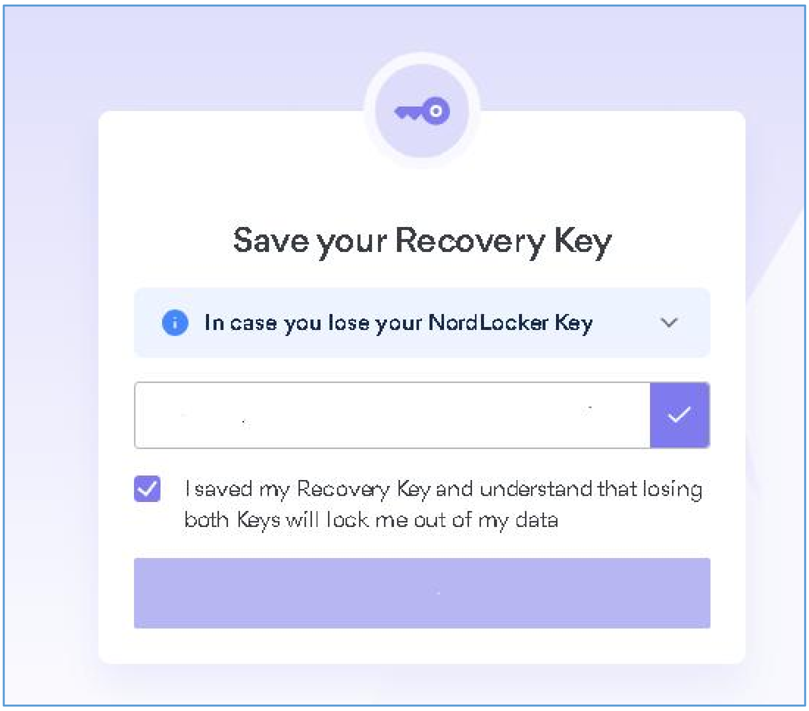
Only you know what your Key is. We don't store it. Memorize it and don't forget it.

Create NordLocker Key

Repeat NordLocker Key

Use a minimum of 6 characters

4. **Encrypt Files:**
 - Use NordLocker to encrypt your sensitive files and folders.
 - Follow the application's interface to select and encrypt the desired files.

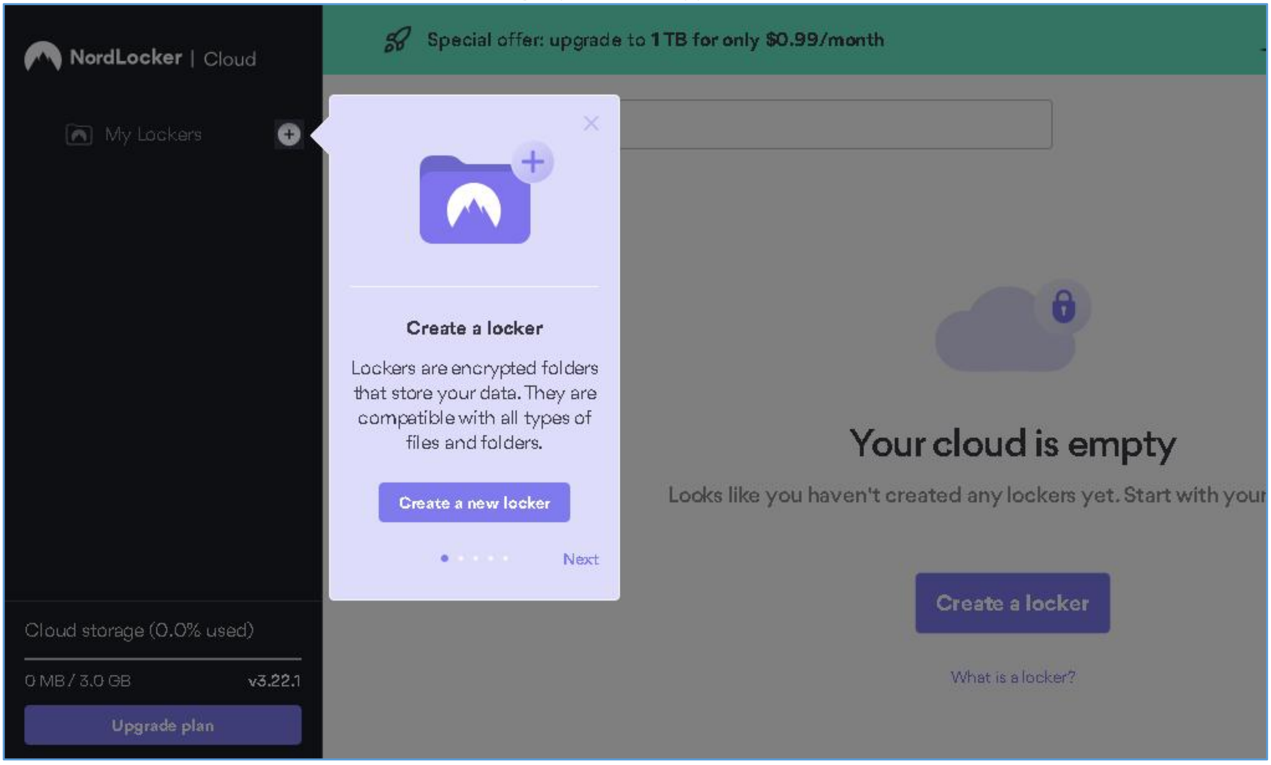


Save your Recovery Key

In case you lose your NordLocker Key

☑ I saved my Recovery Key and understand that losing both Keys will lock me out of my data

5. **Organize Encrypted Files:**
  - Explore the NordLocker interface to organize your encrypted files.
  - Create folders, move files, and manage your encrypted content.

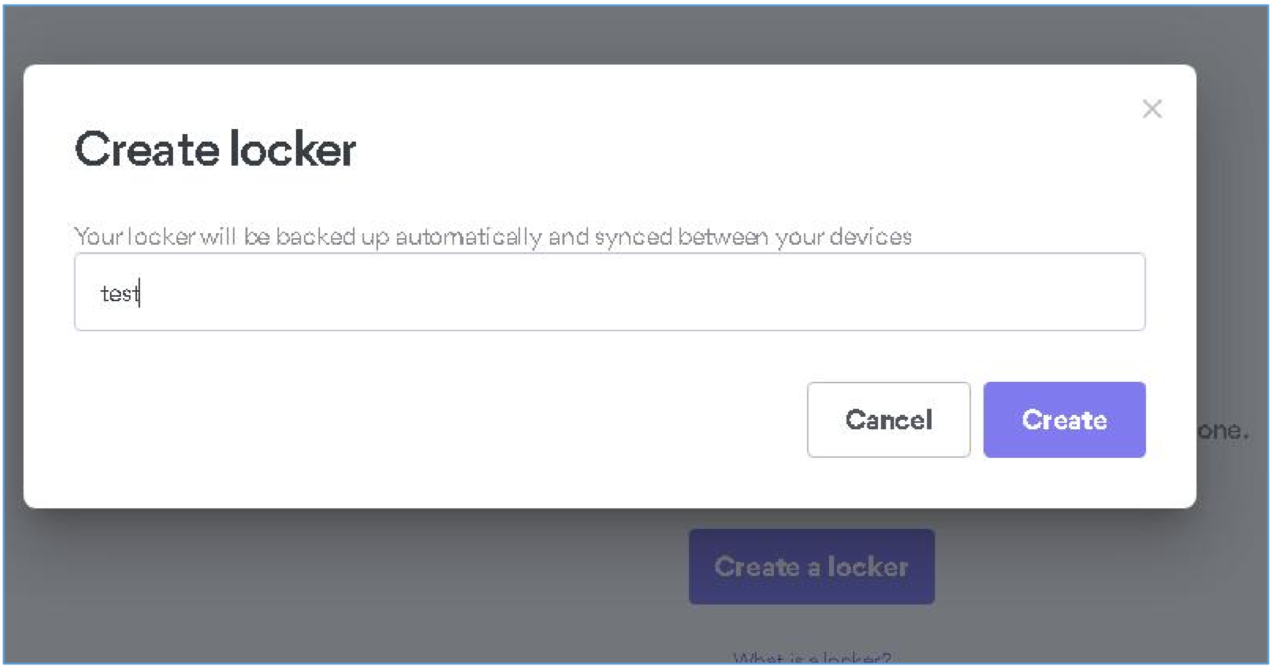


Create a locker

Lockers are encrypted folders that store your data. They are compatible with all types of files and folders.

Your cloud is empty

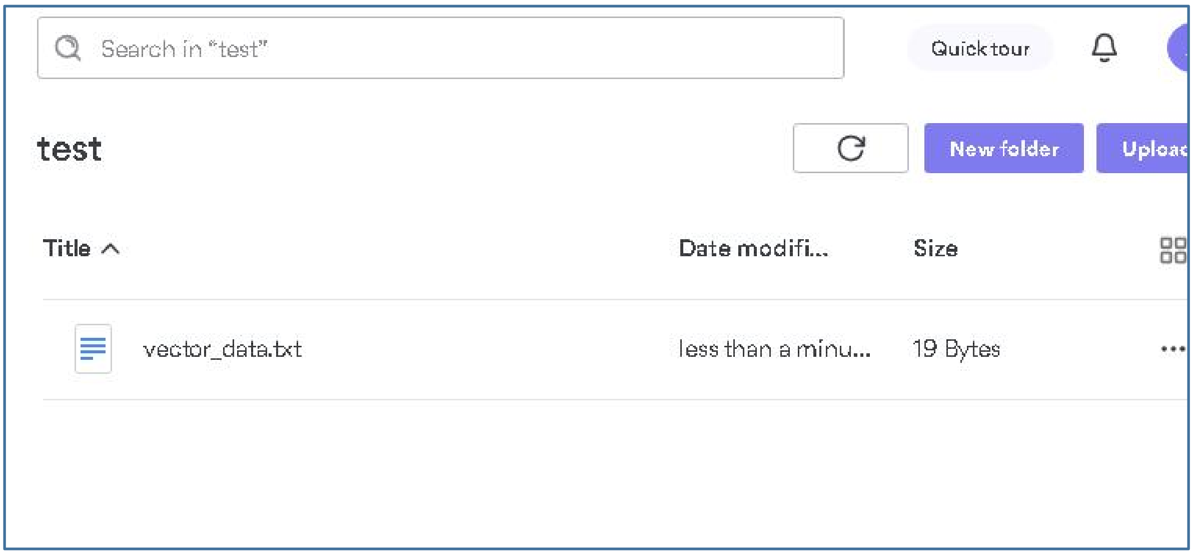
Looks like you haven't created any lockers yet. Start with your



Create locker

Your locker will be backed up automatically and synced between your devices

test



test

Title	Date modifi...	Size
vector_data.txt	less than a minu...	19 Bytes

6. **Access and Decrypt:**
   - Whenever you need to access your encrypted files, open NordLocker.
   - Use your account credentials to log in and decrypt the files you want to use.

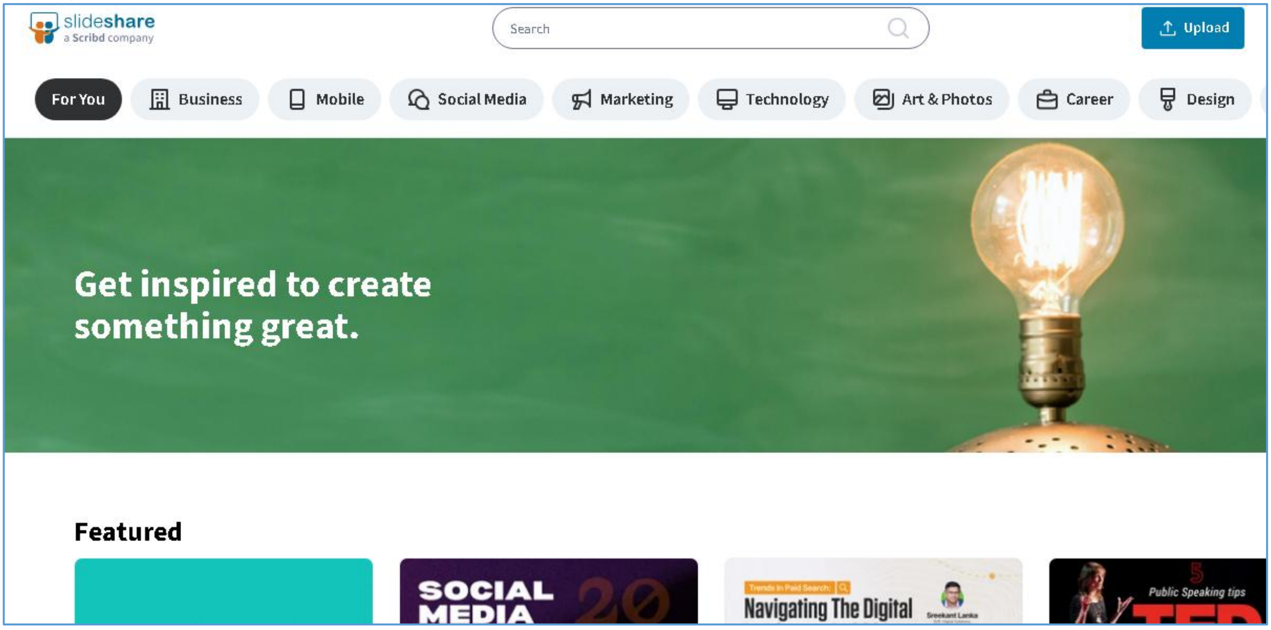
Question 5:

Work with SlideShare (http://www.slideshare.net/) which is a cloud service for slide sharing owned and controlled by LinkedIn.

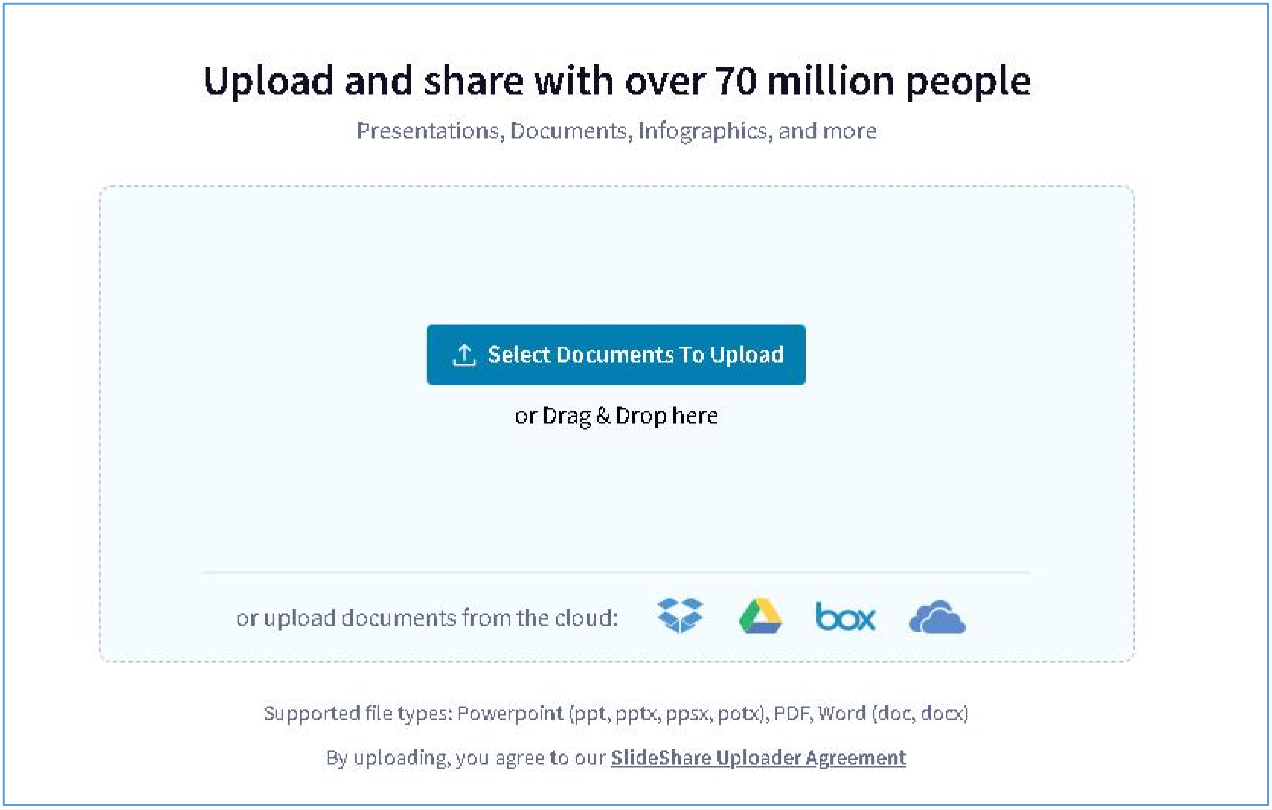
Answer:

SlideShare is a popular online platform for sharing and discovering professional presentations, documents, and infographics. Acquired by LinkedIn in 2012, it serves as a valuable resource for professionals, educators, and businesses to showcase their expertise and insights. Users can upload, share, and embed presentations across various topics, fostering a collaborative environment for knowledge dissemination and networking.

1. Login to SlideShare.



2.Click on upload and select document to upload.


Upload and share with over 70 million people

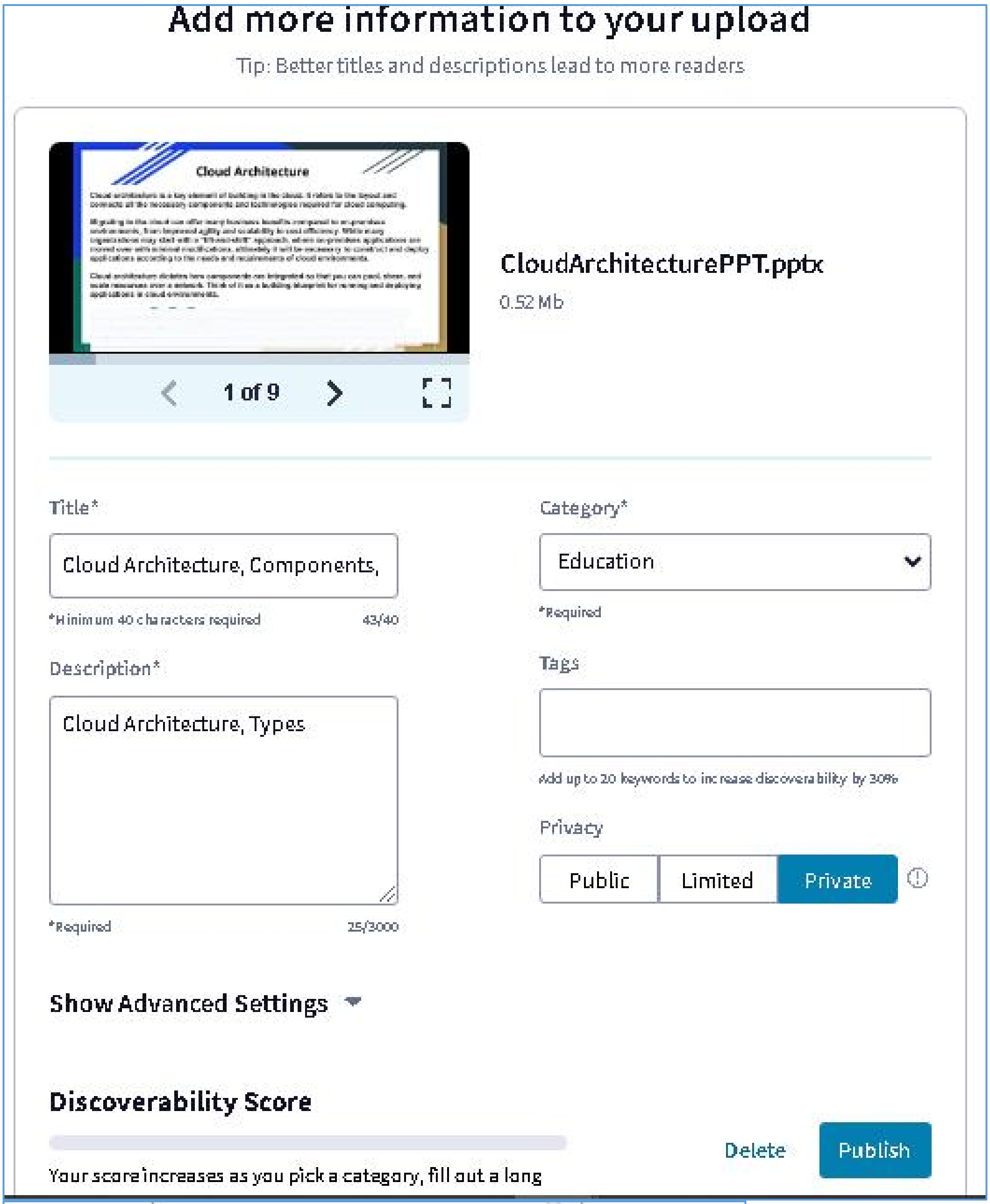
Presentations, Documents, Infographics, and more

Select Documents To Upload
or Drag & Drop here
or upload documents from the cloud:
box

Supported file types: Powerpoint (ppt, pptx, ppsx, potx), PDF, Word (doc, docx)

By uploading, you agree to our SlideShare Uploader Agreement

3.Add information about the document and Publish



Add more information to your upload

Tip: Better titles and descriptions lead to more readers

CloudArchitecturePPT.pptx

0.52 Mb

Title*

Cloud Architecture, Components,

*Minimum 40 characters required 43/40

Category*

Education

*Required

Description*

Cloud Architecture, Types

*Required 25/3000

Tags

Add up to 20 keywords to increase discoverability by 30%

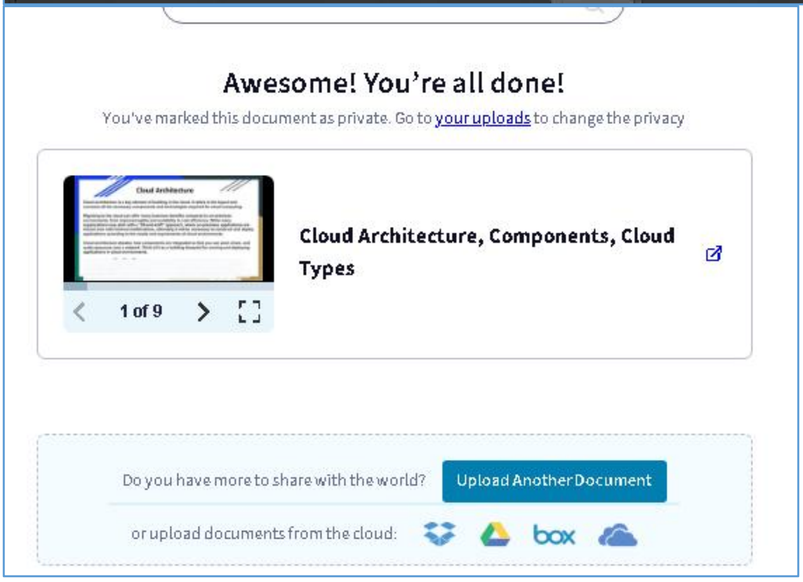
Privacy

Public | Limited | ● Private

Show Advanced Settings

Discoverability Score

Your score increases as you pick a category, fill out a long



Awesome! You're all done!

You've marked this document as private. Go to your uploads to change the privacy

Cloud Architecture, Components, Cloud Types

Do you have more to share with the world? Upload Another Document

or upload documents from the cloud:

Question 6:

Virtualization: Install Oracle Virtual box and create VM on your laptop.

Answer:

Virtualization is a technology that allows multiple operating systems to run on a single physical machine concurrently. By abstracting hardware resources, virtualization enables the creation of virtual machines (VMs), each functioning as an independent and isolated environment. This flexibility in managing and utilizing computing resources enhances efficiency, scalability, and resource utilization in both data centers and personal computing environments.

Steps:

# Install Oracle VirtualBox

## Download Oracle VirtualBox

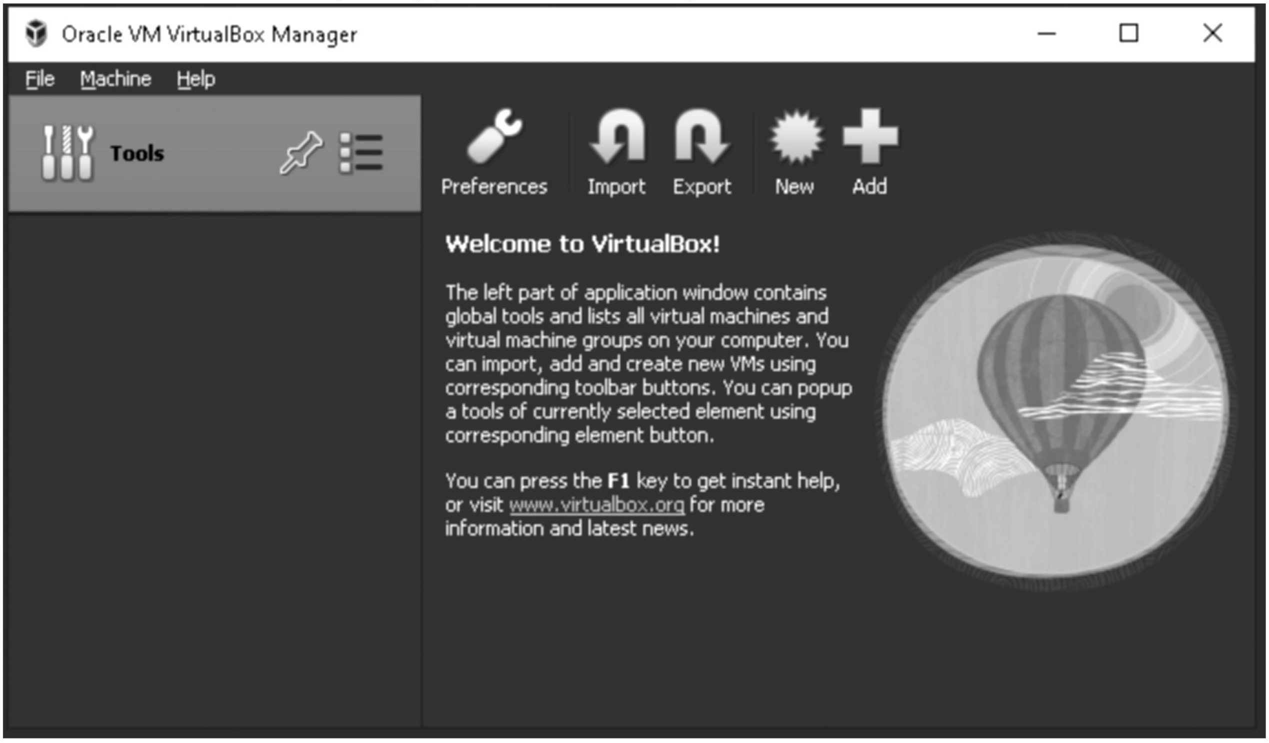
- Visit [Oracle VirtualBox website](https://www.virtualbox.org/).
- Click on "Downloads" and select the version for your OS.
- Download and run the installer.

## Install Oracle VirtualBox

- Follow the installation wizard, accepting default settings.
- Ensure "VirtualBox USB" and "VirtualBox Networking" features are selected.

Launch Oracle VirtualBox

- Launch VirtualBox from your applications or desktop shortcut.



Welcome to VirtualBox!

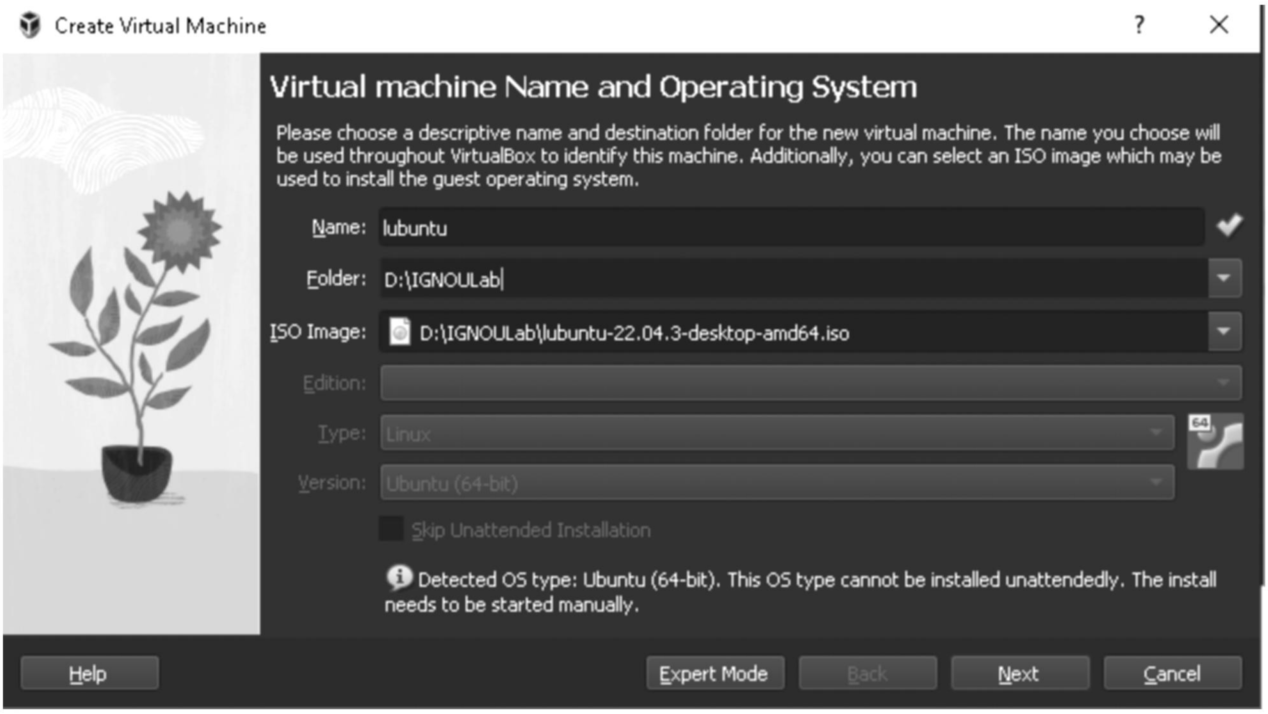
The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can import, add and create new VMs using corresponding toolbar buttons. You can popup a tools of currently selected element using corresponding element button.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for more information and latest news.

# Create a New Virtual Machine for Lubuntu

## Click on "New" to Create a VM

- In VirtualBox Manager, click "New."
- Enter a name for your VM (e.g., "LubuntuVM").



Virtual machine Name and Operating System

Please choose a descriptive name and destination folder for the new virtual machine. The name you choose will be used throughout VirtualBox to identify this machine. Additionally, you can select an ISO image which may be used to install the guest operating system.

Name: lubuntu

Folder: D:\IGNOULab

ISO Image: D:\IGNOULab\lubuntu-22.04.3-desktop-amd64.iso

Edition:

Type: Linux

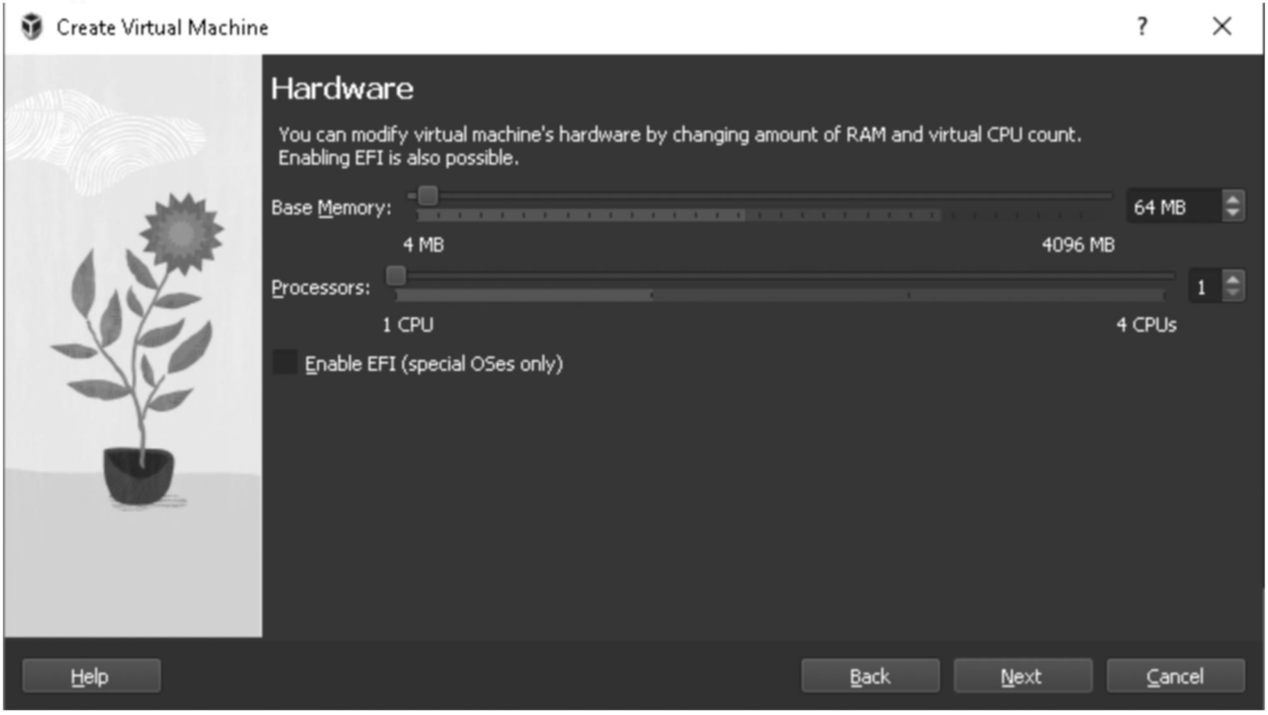
Version: Ubuntu (64-bit)

□ Skip Unattended Installation

Detected OS type: Ubuntu (64-bit). This OS type cannot be installed unattendedly. The install needs to be started manually.

## Choose Operating System and Version
- Select "Linux" as the type.
- Choose "Ubuntu" as the version (Lubuntu is based on Ubuntu).

Allocate Memory (RAM)
- Assign 1GB to 2GB of RAM for Lubuntu.



Hardware

You can modify virtual machine's hardware by changing amount of RAM and virtual CPU count. Enabling EFI is also possible.

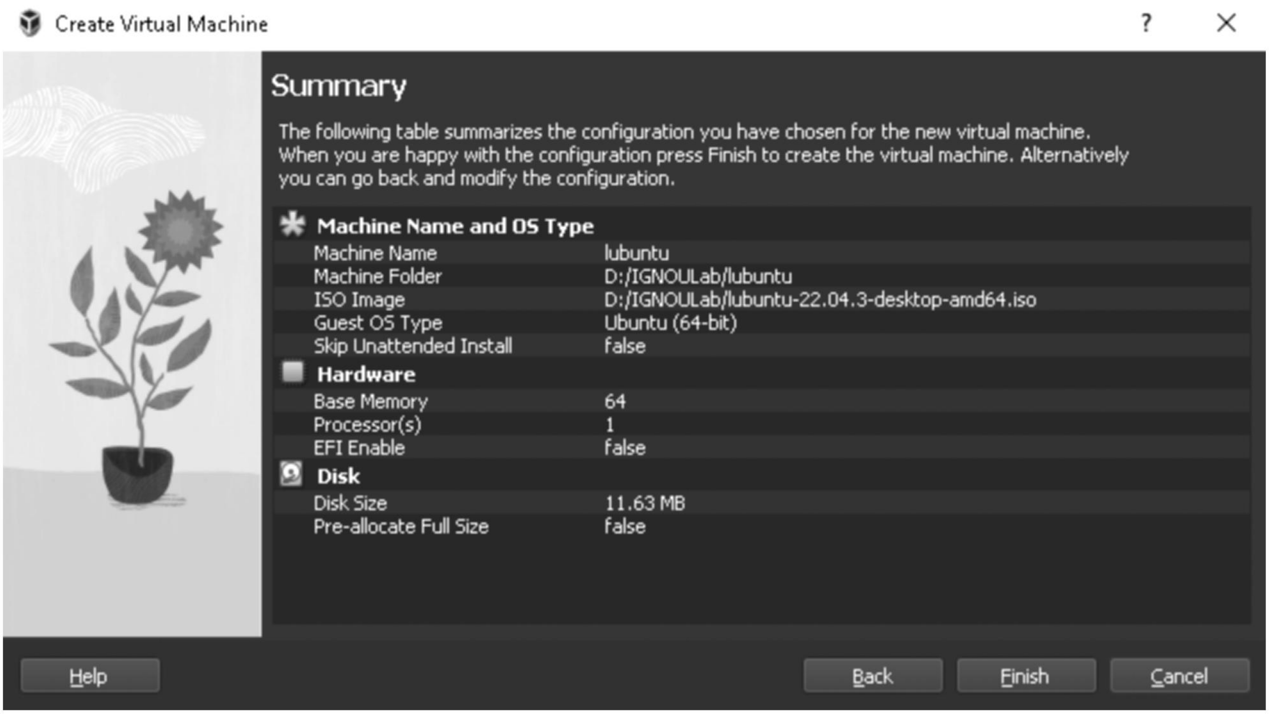
Base Memory: 64 MB

Processors: 1

□ Enable EFI (special OSes only)

## Create a Virtual Hard Disk
- Choose to create a virtual hard disk now.
- Select file type and choose dynamically allocated or fixed size storage.

Set Virtual Hard Disk Size
- Specify at least 10GB for Lubuntu.



Summary

The following table summarizes the configuration you have chosen for the new virtual machine. When you are happy with the configuration press Finish to create the virtual machine. Alternatively you can go back and modify the configuration.

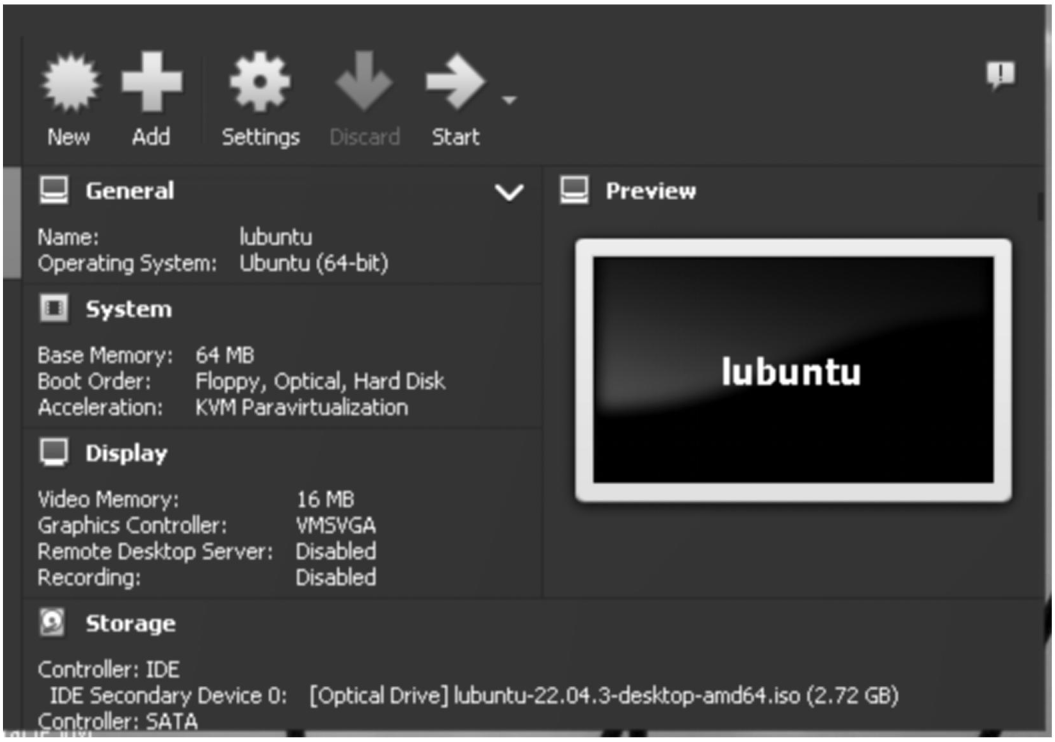
Machine Name and OS Type	
Machine Name	lubuntu
Machine Folder	D:/IGNOULab/lubuntu
ISO Image	D:/IGNOULab/lubuntu-22.04.3-desktop-amd64.iso
Guest OS Type	Ubuntu (64-bit)
Skip Unattended Install	false
Hardware	
Base Memory	64
Processor(s)	1
EFI Enable	false
Disk	
Disk Size	11.63 MB
Pre-allocate Full Size	false

## Mount Lubuntu ISO

- In VirtualBox Manager, select your VM and click "Settings."
- Under "System," move the virtual optical disk up in the boot order.
- Under "Storage," select the empty disk icon and choose the Lubuntu installation ISO file.

Start the Virtual Machine

- With the Lubuntu VM selected, click "Start" in VirtualBox Manager.
- Follow on-screen instructions to install Lubuntu on the VM.



General

Name: lubuntu
Operating System: Ubuntu (64-bit)

System

Base Memory: 64 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: KVM Paravirtualization

Display

Video Memory: 16 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage

Controller: IDE
IDE Secondary Device 0: [Optical Drive] lubuntu-22.04.3-desktop-amd64.iso (2.72 GB)
Controller: SATA

## Complete Lubuntu Installation

- Install Lubuntu, configure settings, and follow installation prompts.
- Remove the ISO from the virtual optical drive after installation.

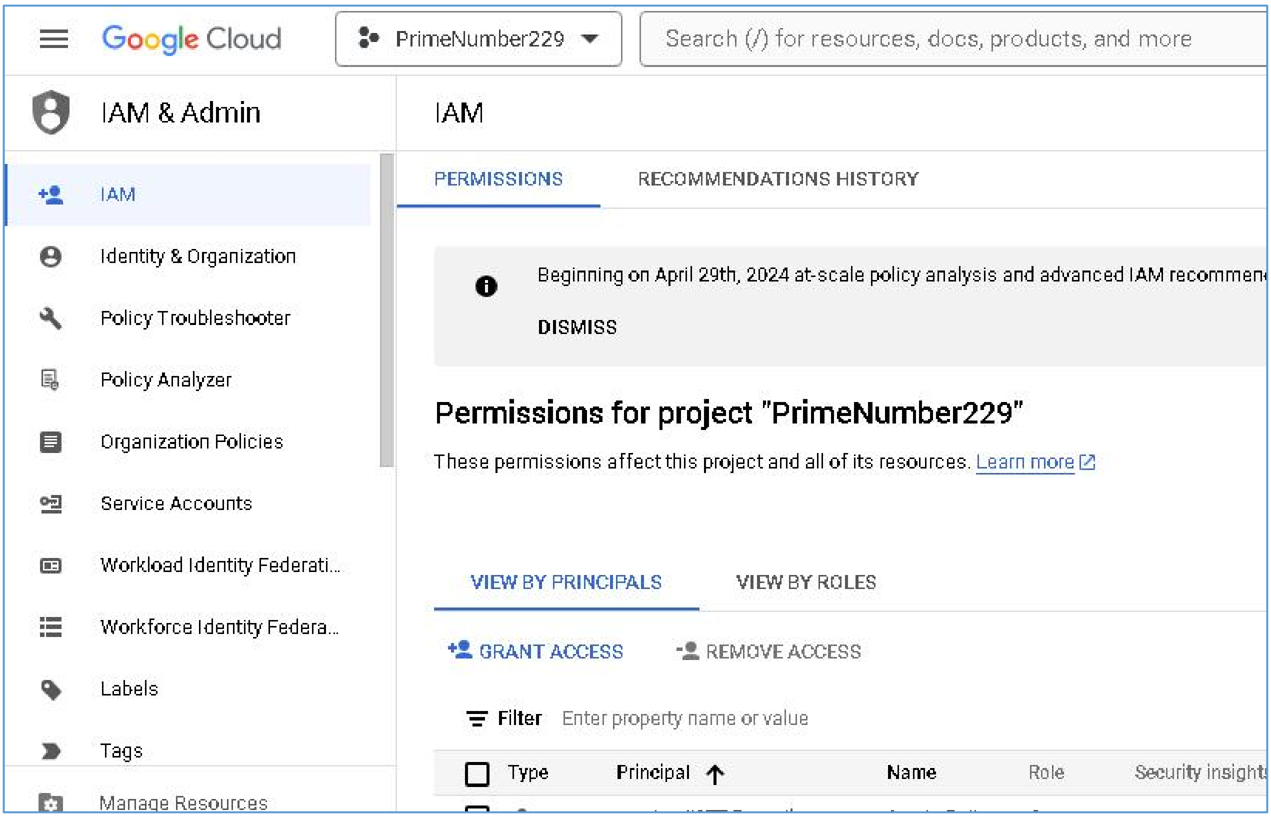
Question 7:

Establish an Google Cloud Platform(https://cloud.google.com) account (use trail version). Explore the following:

(i) IAM & Admin
(ii) Billing
(iii) Marketplace (Creating Virtual Machines)
(iv) Compute Engine
(v) Cloud Storage

Answer:

(i) IAM & Admin



IAM

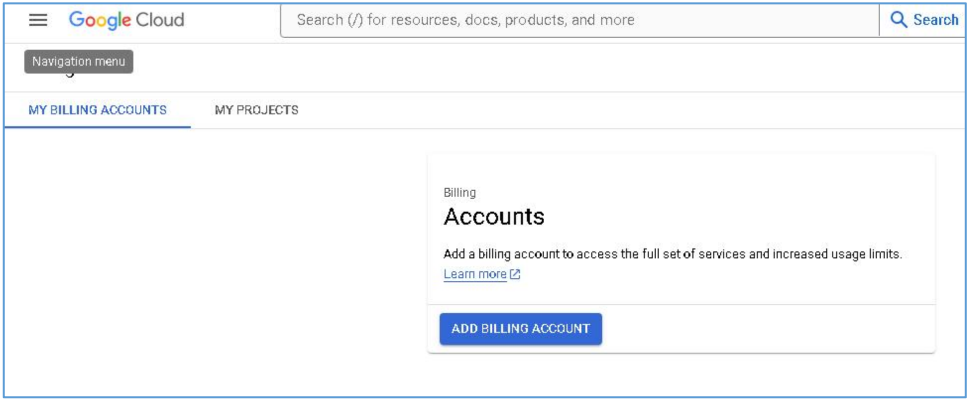
Beginning on April 29th, 2024 at-scale policy analysis and advanced IAM recommen

Permissions for project "PrimeNumber229"

These permissions affect this project and all of its resources. Learn more

□	Type	Principal ↑	Name	Role	Security insight

(ii) Billing

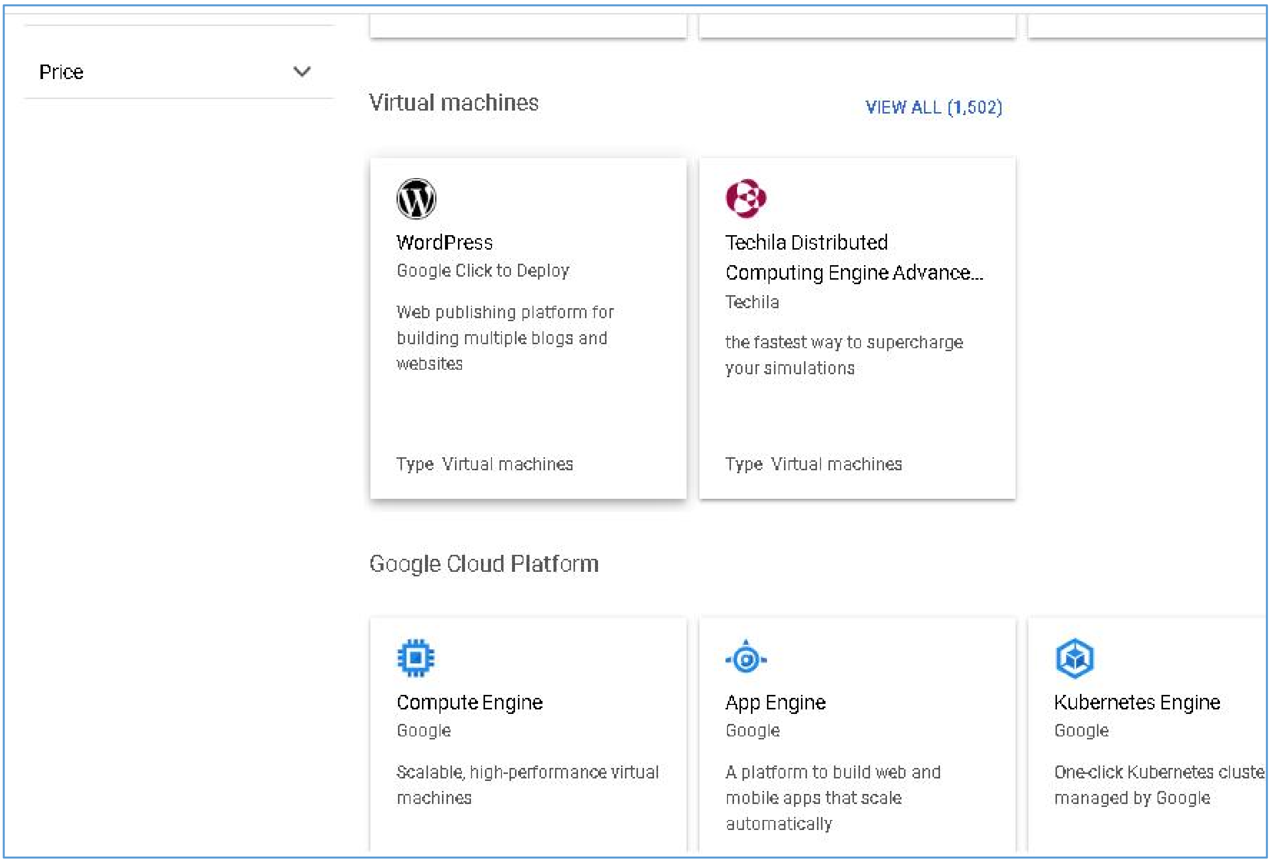


Billing

Accounts

Add a billing account to access the full set of services and increased usage limits.
Learn more

(iii) Marketplace (Creating Virtual Machines) and
(iv) Compute Engine



Price

Virtual machines

VIEW ALL (1,502)

WordPress
Google Click to Deploy
Web publishing platform for building multiple blogs and websites
Type Virtual machines

Techila Distributed Computing Engine Advance...
Techila
the fastest way to supercharge your simulations
Type Virtual machines

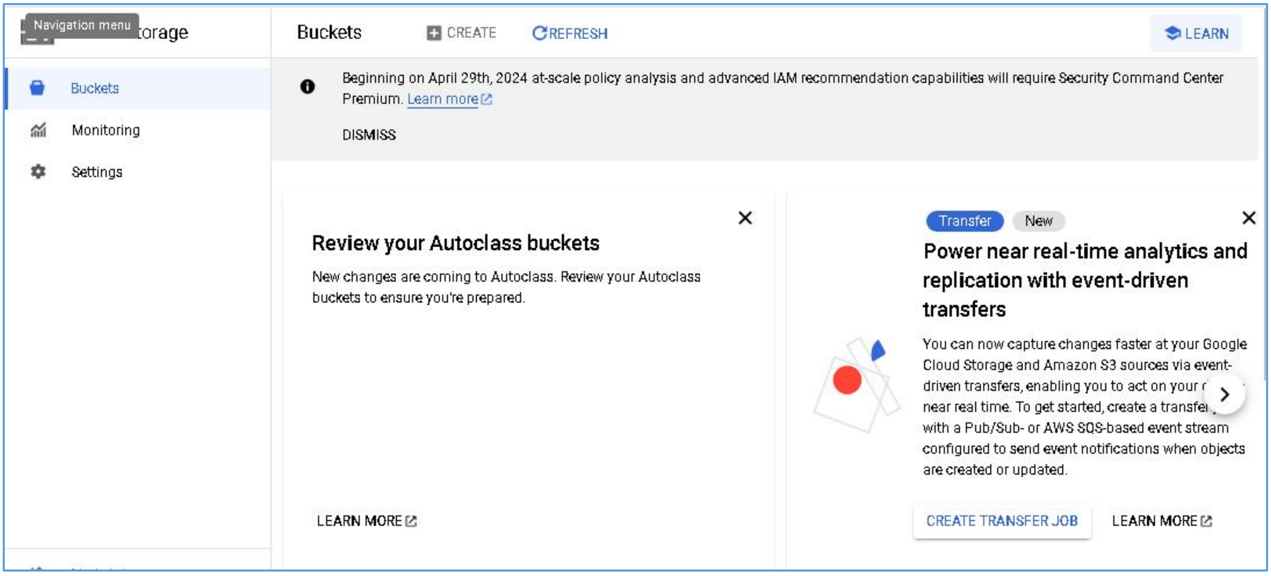
Google Cloud Platform

Compute Engine
Google
Scalable, high-performance virtual machines

App Engine
Google
A platform to build web and mobile apps that scale automatically

Kubernetes Engine
Google
One-click Kubernetes cluste managed by Google

(v) Cloud Storage



Navigation menu torage

Buckets

Monitoring

Settings

Buckets CREATE REFRESH LEARN

Beginning on April 29th, 2024 at-scale policy analysis and advanced IAM recommendation capabilities will require Security Command Center Premium. Learn more

DISMISS

Review your Autoclass buckets

New changes are coming to Autoclass. Review your Autoclass buckets to ensure you're prepared.

LEARN MORE

Transfer New

Power near real-time analytics and replication with event-driven transfers

You can now capture changes faster at your Google Cloud Storage and Amazon S3 sources via event-driven transfers, enabling you to act on your near real time. To get started, create a transfe with a Pub/Sub- or AWS SQS-based event stream configured to send event notifications when objects are created or updated.

CREATE TRANSFER JOB LEARN MORE

Question 8:

Create a list of cloud services provided by AWS. List the steps to set up an Elastic Compute Cloud (EC2) instance. Do these services lower the cost of operation of an organization? Justify your answer.

Answer:

List of cloud services provided by AWS:

1. Elastic Compute Cloud (EC2)
2. Simple Storage Service (S3)
3. Relational Database Service (RDS)
4. Lambda
5. Elastic Beanstalk
6. CloudFront
7. DynamoDB
8. Elastic Load Balancing (ELB)
9. Elastic Container Service (ECS)
10. Virtual Private Cloud (VPC)
11. Simple Notification Service (SNF)
12. Simple Queue Service (SQS)
13. Glacier
14. Route 53
15. CloudWatch
16. Identity and Access Management (IAM)

List steps to set up an Elastic Compute Cloud (EC2) instance:
Complete the tasks in this section to get set up for launching an Amazon EC2 instance for the first time:

1. Sign up for an AWS account
2. Create an administrative user
3. Create a key pair
4. Create a security group

1. Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one. To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

2. Create an administrative user

After you sign up for an AWS account, create an administrative user so that you don't use the root user for everyday tasks. Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing Root user and entering your AWS account email address. On the next page, enter your password.
For help signing in by using root user, see Signing in as the root user in the AWS Sign-In User Guide.
2. Turn on multi-factor authentication (MFA) for your root user.
For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the IAM User Guide.

Create an administrative user

- For your daily administrative tasks, grant administrative access to an administrative user in AWS IAM Identity Center (successor to AWS Single Sign-On).

For instructions, see Getting started in the AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide.

Sign in as the administrative user

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the AWS Sign-In User Guide.

3. Create a key pair

To create your key pair

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigaton pane, choose Key Pairs.
3. Choose Create key pair.
4. For Name, enter a descriptive name for the key pair. Amazon EC2 associates the public key with the name that you specify as the key name. A key name can include up to 255 ASCII characters. It can’t include leading or trailing spaces.
5. For Key pair type, choose either RSA or ED25519. Note that ED25519 keys are not supported for Windows instances.
6. For Private key file format, choose the format in which to save the private key. To save the private key in a format that can be used with OpenSSH, choose pem. To save the private key in a format that can be used with PuTTY, choose ppk.
7. Choose Create key pair.

8. The private key file is automatically downloaded by your browser. The base file name is the name you specified as the name of your key pair, and the file name extension is determined by the file format you chose. Save the private key file in a safe place.
9. If you plan to use an SSH client on a macOS or Linux computer to connect to your Linux instance, use the following command to set the permissions of your private key file so that only you can read it.

4. Create a security group

Security groups act as a firewall for associated instances, controlling both inbound and outbound traffic at the instance level. You must add rules to a security group that enable you to connect to your instance from your IP address using SSH. You can also add rules that allow inbound and outbound HTTP and HTTPS access from anywhere.

Note that if you plan to launch instances in multiple AWS Regions, you'll need to create a security group in each Region. For more information about Regions, see Regions and Zones.

Do these services lower the cost of operation of an organisation?

Yes, these services can lower the cost of operation for an organization. AWS provides a variety of services that can help organizations save on infrastructure and operational costs. For example, using EC2 instances instead of physical servers can eliminate the need for organizations to purchase and maintain their own hardware. Additionally, services like S3 and Glacier can help organizations save on storage costs by providing a scalable, cost-effective way to store data. Overall, AWS provides a flexible, scalable, and cost-effective way for organizations to meet their IT needs.

Question 9:

Use Google App Engine to write a Google app engine program to generate prime numbers up to a number given number n and deploy it to Google cloud.

Answer:

To write a Google App Engine program to generate prime numbers up to a given number n, we can use Python programming language and the Flask web framework. Here’s how to do it:

1. Create a new project on the Google Cloud Platform Console, or use an existing one.
2. Install the Google Cloud SDK on your local machine.
3. Create a new virtual environment and activate it using the following commands:

```
python -m venv myenv
source myenv/bin/activate
```

Install Flask using the following command:

```
        pip install Flask
```

Program:

```
import webapp2
class MainHandler(webapp2.RequestHandler):
        def get(self):
                n = int(self.request.get('n'))
                 primes = []
                for i in range(2, n+1):
                is_prime = True
for j in range(2, int(i”0.5)+1): if i % j == 0:
is prime = False break
        if is prime:
                primes.append(i)
        self.response.write(' '.join(str(p) for p in primes))

app = webapp2.WSGIApplication([
        ('/', MainHandler)
], debug=True)
```

Deploy it to Google cloud: Deploy the application to Google App Engine using the following command: gcloud app deploy.

This command will upload your application code to Google Cloud Platform and deploy it to the App Engine. Once the deployment is complete, you can access your application by visiting https://[YOUR_PROJECT ID].appspot.com/primes?n=100 in your web browser, where [YOUR_PROJECT_ID] is your Google Cloud project ID and n=100 is an example query parameter indicating that you want to generate all prime numbers up to 100

Section 2: Data Science Lab

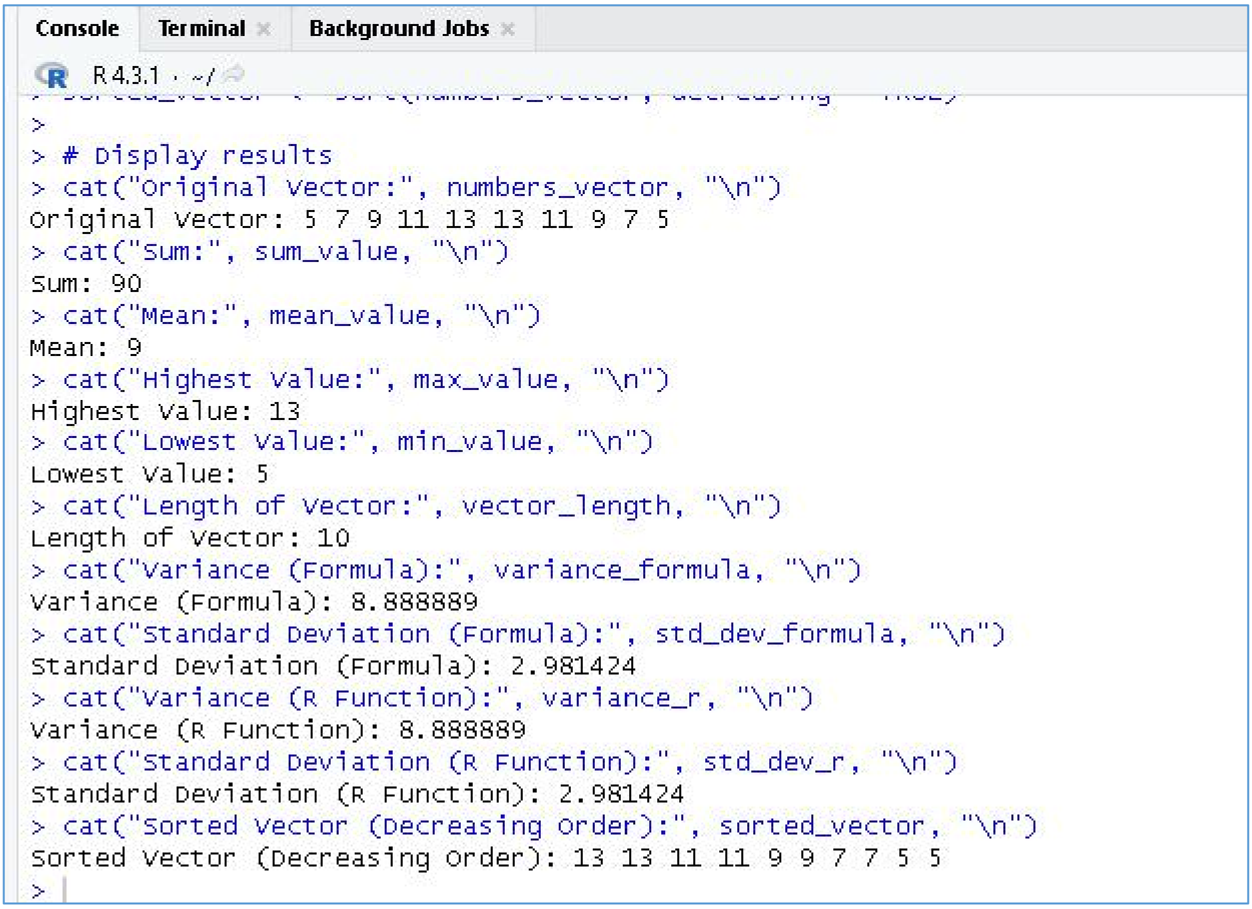
Question 1:

Create a vector of size 10, having the values 5,7,9,11,13,13,11,9,7,5. Compute the sum, mean, highest and lowest of these values. Compute the length of this vector? Find the variance and standard deviation for the data of this vector, using the formula for variance and standard deviation. Compare these values by computing the variance and standard deviation using R function. Sort this array values in decreasing order.

Program:

```
# Create a vector
numbers_vector <- c(5, 7, 9, 11, 13, 13, 11, 9, 7, 5)

# Compute the sum, mean, highest, and lowest
sum_value <- sum(numbers_vector)
mean_value <- mean(numbers_vector)
max_value <- max(numbers_vector)
min_value <- min(numbers_vector)

# Compute the length of the vector
vector_length <- length(numbers_vector)

# Compute variance and standard deviation using formulas
variance_formula <- sum((numbers_vector - mean_value)^2) / (vector_length - 1)
std_dev_formula <- sqrt(variance_formula)

# Compute variance and standard deviation using R functions
variance_r <- var(numbers_vector)
std_dev_r <- sd(numbers_vector)

# Sort the vector in decreasing order
sorted_vector <- sort(numbers_vector, decreasing = TRUE)

# Display results
cat("Original Vector:", numbers_vector, "\n")
cat("Sum:", sum_value, "\n")
cat("Mean:", mean_value, "\n")
cat("Highest Value:", max_value, "\n")
cat("Lowest Value:", min_value, "\n")
cat("Length of Vector:", vector_length, "\n")
cat("Variance (Formula):", variance_formula, "\n")
cat("Standard Deviation (Formula):", std_dev_formula, "\n")
cat("Variance (R Function):", variance_r, "\n")
cat("Standard Deviation (R Function):", std_dev_r, "\n")
cat("Sorted Vector (Decreasing Order):", sorted_vector, "\n")
```

Output:



```
> sorted_vector <- sort(numbers_vector, decreasing = TRUE)
>
> # Display results
> cat("Original Vector:", numbers_vector, "\n")
Original Vector: 5 7 9 11 13 13 11 9 7 5
> cat("Sum:", sum_value, "\n")
Sum: 90
> cat("Mean:", mean_value, "\n")
Mean: 9
> cat("Highest Value:", max_value, "\n")
Highest Value: 13
> cat("Lowest Value:", min_value, "\n")
Lowest Value: 5
> cat("Length of Vector:", vector_length, "\n")
Length of Vector: 10
> cat("Variance (Formula):", variance_formula, "\n")
Variance (Formula): 8.888889
> cat("Standard Deviation (Formula):", std_dev_formula, "\n")
Standard Deviation (Formula): 2.981424
> cat("Variance (R Function):", variance_r, "\n")
Variance (R Function): 8.888889
> cat("Standard Deviation (R Function):", std_dev_r, "\n")
Standard Deviation (R Function): 2.981424
> cat("Sorted Vector (Decreasing Order):", sorted_vector, "\n")
Sorted Vector (Decreasing Order): 13 13 11 11 9 9 7 7 5 5
>
```

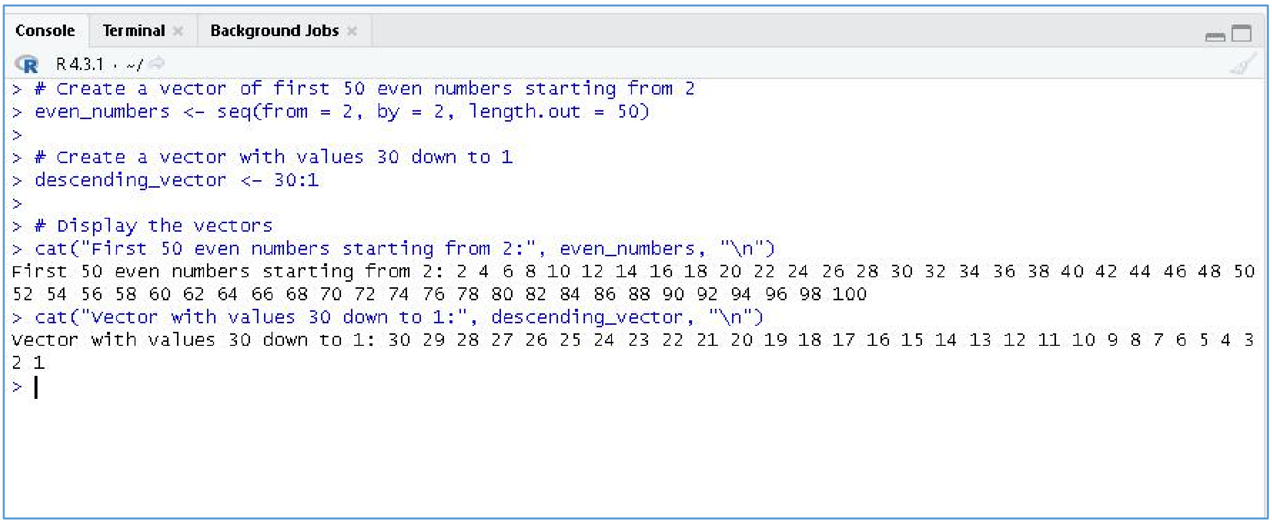
Question 2:

Create a vector of first 50 even numbers, starting from 2. Also create a vector having values 30 down to 1, as 30, 29, ...,1

Program:

```
# Create a vector of first 50 even numbers starting from 2
even_numbers <- seq(from = 2, by = 2, length.out = 50)

# Create a vector with values 30 down to 1
descending_vector <- 30:1

# Display the vectors
cat("First 50 even numbers starting from 2:", even_numbers, "\n")
cat("Vector with values 30 down to 1:", descending_vector, "\n")
```

Output:



```
> # Create a vector of first 50 even numbers starting from 2
> even_numbers <- seq(from = 2, by = 2, length.out = 50)
>
> # Create a vector with values 30 down to 1
> descending_vector <- 30:1
>
> # Display the vectors
> cat("First 50 even numbers starting from 2:", even_numbers, "\n")
First 50 even numbers starting from 2: 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98 100
> cat("Vector with values 30 down to 1:", descending_vector, "\n")
Vector with values 30 down to 1: 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3
2 1
> |
```

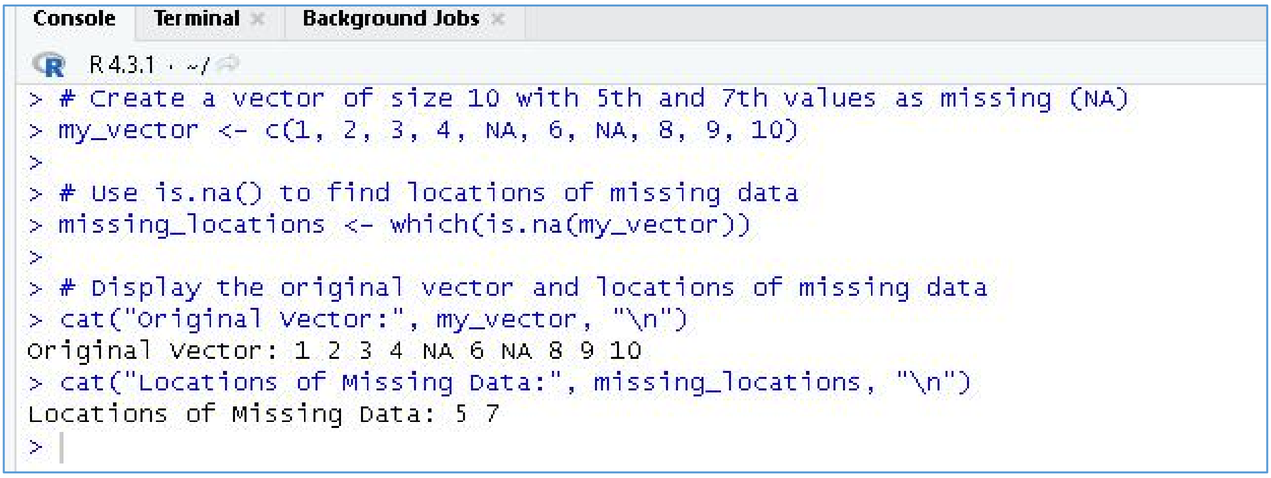
Question 3:

Create a vector of size 10 with 5th and 7th values as missing (store these values as NA). Use the "is.na()" to find locations of missing data.

Program:

```
# Create a vector of size 10 with 5th and 7th values as missing (NA)
my_vector <- c(1, 2, 3, 4, NA, 6, NA, 8, 9, 10)

# Use is.na() to find locations of missing data
missing_locations <- which(is.na(my_vector))

# Display the original vector and locations of missing data
cat("Original Vector:", my_vector, "\n")
cat("Locations of Missing Data:", missing_locations, "\n")
```

Output:



```
> # Create a vector of size 10 with 5th and 7th values as missing (NA)
> my_vector <- c(1, 2, 3, 4, NA, 6, NA, 8, 9, 10)
>
> # Use is.na() to find locations of missing data
> missing_locations <- which(is.na(my_vector))
>
> # Display the original vector and locations of missing data
> cat("Original Vector:", my_vector, "\n")
Original Vector: 1 2 3 4 NA 6 NA 8 9 10
> cat("Locations of Missing Data:", missing_locations, "\n")
Locations of Missing Data: 5 7
> 
```

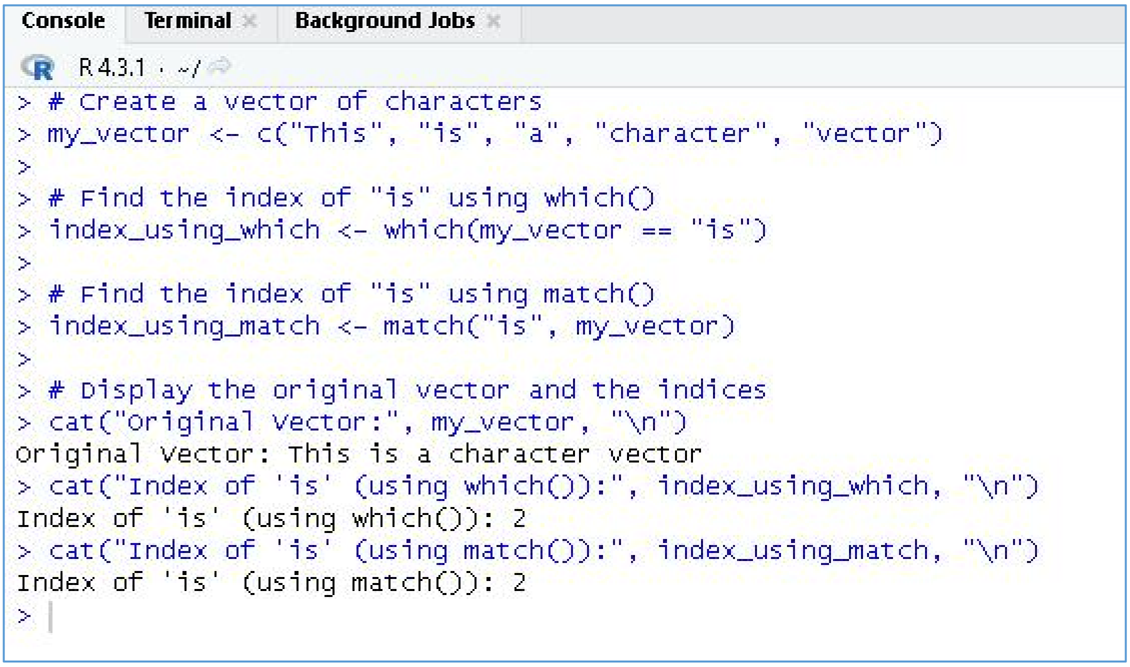
Question 4:

Create a vector of characters of size 5, consisting of values: "This" "is" "a" "character" "vector". Find the index of value "is" in the vector using which() or match().

Program:

```
# Create a vector of characters
my_vector <- c("This", "is", "a", "character", "vector")

# Find the index of "is" using which()
index_using_which <- which(my_vector == "is")

# Find the index of "is" using match()
index_using_match <- match("is", my_vector)

# Display the original vector and the indices
cat("Original Vector:", my_vector, "\n")
cat("Index of 'is' (using which()):", index_using_which, "\n")
cat("Index of 'is' (using match()):", index_using_match, "\n")
```

Output:



```
> # Create a vector of characters
> my_vector <- c("This", "is", "a", "character", "vector")
>
> # Find the index of "is" using which()
> index_using_which <- which(my_vector == "is")
>
> # Find the index of "is" using match()
> index_using_match <- match("is", my_vector)
>
> # Display the original vector and the indices
> cat("Original Vector:", my_vector, "\n")
Original Vector: This is a character vector
> cat("Index of 'is' (using which()):", index_using_which, "\n")
Index of 'is' (using which()): 2
> cat("Index of 'is' (using match()):", index_using_match, "\n")
Index of 'is' (using match()): 2
>
```

Question 5:

It is always good to store numerical values rather than textual data. However, while input or output the textual values are easier to understand. An example, for this is as follows in R:

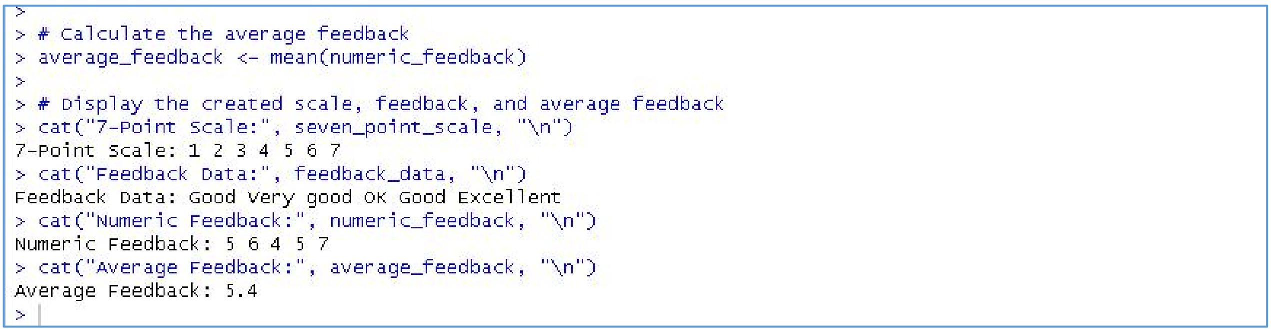
```
> Fivepointscale=c(1:5)

> names(Fivepointscale) = c("Not Satisfactory", "Satisfactory", "Fair", "Good", "Very Good")

> Feedback = Fivepointscale[c("Good", Satisfactory ")J
```

Create a 7-point scale of information input and use this scale to input feedback of 5 students about a question like "Feedback of experience of using an application (Bad, Somewhat bad, not good, ok, good, very good, excellent)". Find the average of the feedback.

Program:

```
# Create a 7-point scale
seven_point_scale <- c(1:7)
names(seven_point_scale) <- c("Bad", "Somewhat bad", "Not good", "OK", "Good", "Very good",
"Excellent")

# Input feedback for 5 students
feedback_data <- c("Good", "Very good", "OK", "Good", "Excellent")

# Convert feedback to numerical values using the scale
numeric_feedback <- seven_point_scale[feedback_data]

# Calculate the average feedback
average_feedback <- mean(numeric_feedback)

# Display the created scale, feedback, and average feedback
cat("7-Point Scale:", seven_point_scale, "\n")
cat("Feedback Data:", feedback_data, "\n")
cat("Numeric Feedback:", numeric_feedback, "\n")
cat("Average Feedback:", average_feedback, "\n")
```

Output:



```
>
> # Calculate the average feedback
> average_feedback <- mean(numeric_feedback)
>
> # Display the created scale, feedback, and average feedback
> cat("7-Point Scale:", seven_point_scale, "\n")
7-Point Scale: 1 2 3 4 5 6 7
> cat("Feedback Data:", feedback_data, "\n")
Feedback Data: Good Very good OK Good Excellent
> cat("Numeric Feedback:", numeric_feedback, "\n")
Numeric Feedback: 5 6 4 5 7
> cat("Average Feedback:", average_feedback, "\n")
Average Feedback: 5.4
>
```

Question 6:

Create or download sample data of customers of an e-commerce website. Consider it has factors like family income, total amount spent last month by the customer, Is subscriber of product review pages, etc. Classify the customers into the following categories:

High spenders, medium spenders, Low spenders.

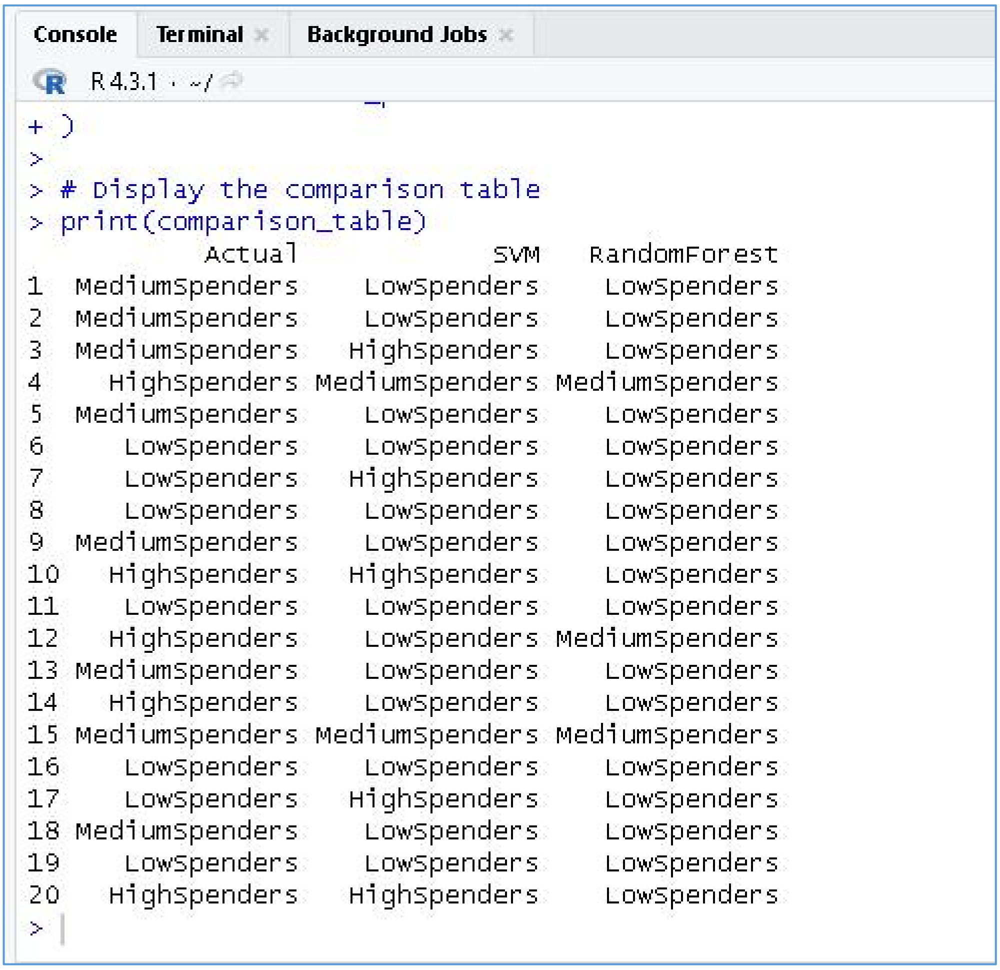
You may use any two classification algorithms/techniques and compare the results of the two classifiers.

Program:

```
# Install readxl library for Excel file reading
install.packages("readxl")

# Install e1071 library for support vector machines and other machine learning algorithms
install.packages("e1071")

# Install randomForest library for building and analyzing random forests
install.packages("randomForest")

library(readxl)
library(e1071)
library(randomForest)

#Read the Excel file into R
customer_data <- read_excel("D:/IGNOULab/customer_data.xlsx", sheet = "customer_data")

# Split the data into training and testing sets
set.seed(123)
split_index <- sample(1:nrow(customer_data), 0.8 * nrow(customer_data))
train_data <- customer_data[split_index, ]
test_data <- customer_data[-split_index, ]

# Train two classification algorithms

# Convert SpendingCategory to a factor with levels
train_data$SpendingCategory <- factor(train_data$SpendingCategory, levels = c("LowSpenders",
"MediumSpenders", "HighSpenders"))

# Check the levels
levels(train_data$SpendingCategory)
train_data <- na.omit(train_data)

# SVM model
svm_model <- svm(SpendingCategory ~ FamilyIncome + SubscriberReviewPages, data =
train_data)
```

```
# Random Forest model
rf_model <- randomForest(SpendingCategory ~ FamilyIncome + SubscriberReviewPages, data =
train_data)

# Predictions using SVM
svm_predictions <- predict(svm_model, test_data)

# Predictions using Random Forest
rf_predictions <- predict(rf_model, test_data)

# Compare the results
comparison_table <- data.frame(
  Actual = test_data$SpendingCategory,
  SVM = svm_predictions,
  RandomForest = rf_predictions
)

# Display the comparison table
print(comparison_table)
```

Output:



```
+ )
> 
> # Display the comparison table
> print(comparison_table)
           Actual            SVM   RandomForest
1  MediumSpenders    LowSpenders    LowSpenders
2  MediumSpenders    LowSpenders    LowSpenders
3  MediumSpenders   HighSpenders    LowSpenders
4    HighSpenders MediumSpenders MediumSpenders
5  MediumSpenders    LowSpenders    LowSpenders
6     LowSpenders    LowSpenders    LowSpenders
7     LowSpenders   HighSpenders    LowSpenders
8     LowSpenders    LowSpenders    LowSpenders
9  MediumSpenders    LowSpenders    LowSpenders
10   HighSpenders   HighSpenders    LowSpenders
11    LowSpenders    LowSpenders    LowSpenders
12   HighSpenders    LowSpenders MediumSpenders
13 MediumSpenders    LowSpenders    LowSpenders
14   HighSpenders    LowSpenders    LowSpenders
15 MediumSpenders MediumSpenders MediumSpenders
16    LowSpenders    LowSpenders    LowSpenders
17    LowSpenders   HighSpenders    LowSpenders
18 MediumSpenders    LowSpenders    LowSpenders
19    LowSpenders    LowSpenders    LowSpenders
20   HighSpenders   HighSpenders    LowSpenders
> |
```

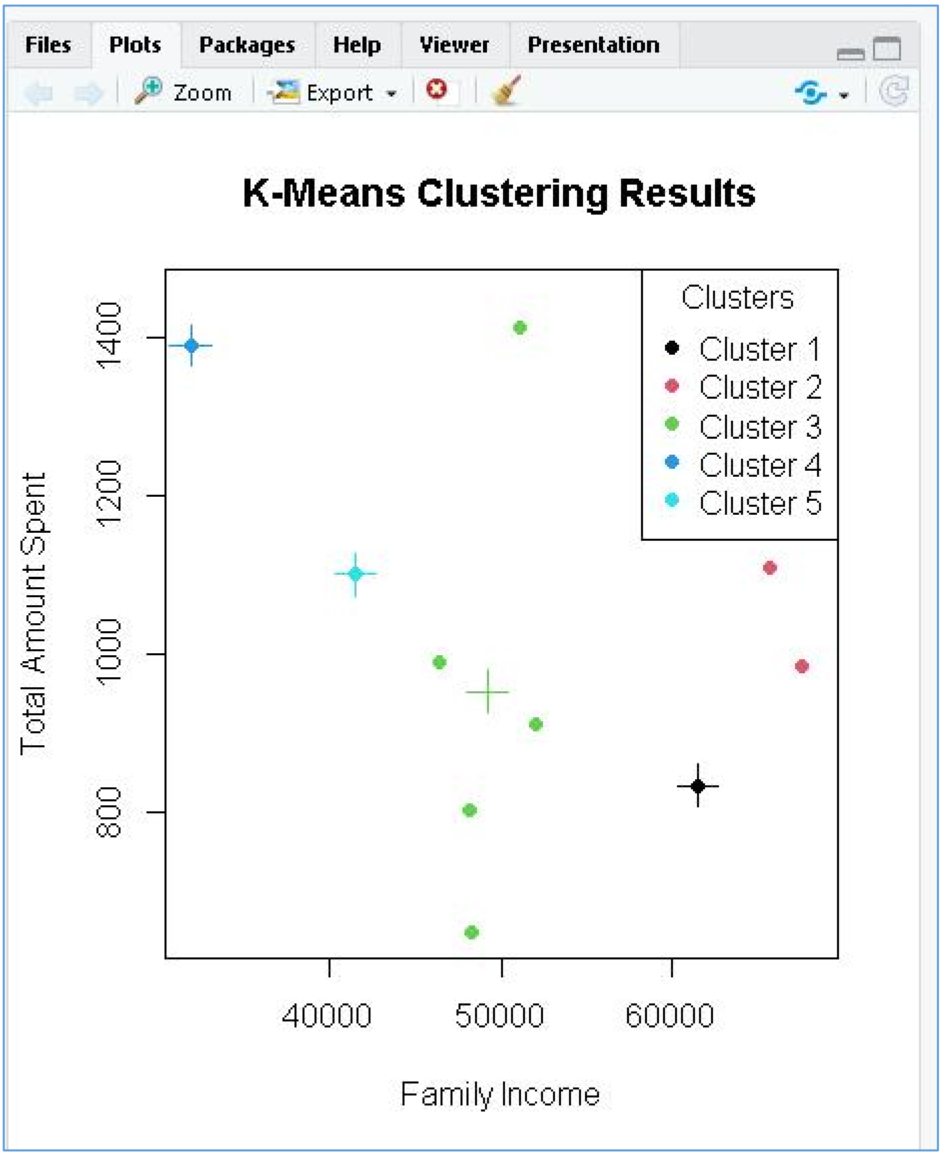
Question 7:

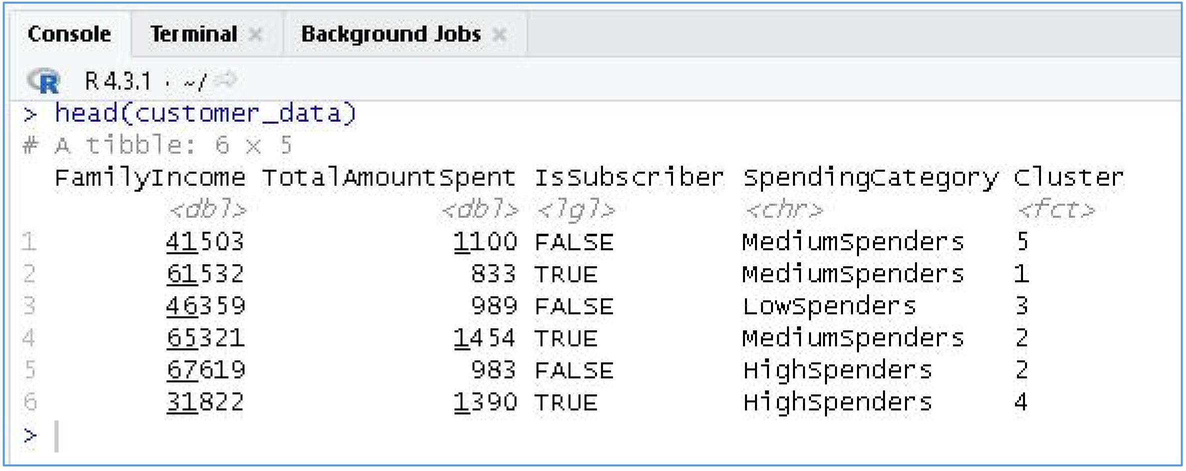
Assuming that the problem, as given above, does not have any categories. Perform k-mean clustering on the data with k = 5

Program:

```
# Install and load required libraries
install.packages("readxl")
library(readxl)

# Load data from Excel file (replace 'your_excel_file.xlsx' with your file path)
customer_data <- read_excel("D:/IGNOULab/customer_data.xlsx", sheet = "customer_data")

# Convert the IsSubscriber column to boolean
customer_data$IsSubscriber <- as.logical(customer_data$IsSubscriber)

# Select relevant columns for clustering
clustering_data <- customer_data[, c("FamilyIncome", "TotalAmountSpent", "IsSubscriber")]

# Check for missing, NaN, or infinite values
clustering_data <- na.omit(clustering_data)

# Set the number of clusters to 5
num_clusters <- 5

# Check if the number of clusters is valid
if (num_clusters < 1 || num_clusters > nrow(clustering_data)) {
  stop("Invalid number of clusters.")
}
# Perform k-means clustering
kmeans_result <- kmeans(clustering_data[, c("FamilyIncome", "TotalAmountSpent")], centers =
num_clusters, nstart = 20)

# Attach the cluster labels to the original data
customer_data$Cluster <- as.factor(kmeans_result$cluster)

# Plot the clusters (2D scatter plot using actual values)
plot(clustering_data$FamilyIncome, clustering_data$TotalAmountSpent, col =
kmeans_result$cluster, pch = 19, main = "K-Means Clustering Results", xlab = "Family Income",
ylab = "Total Amount Spent")

# Add cluster centers to the plot
points(kmeans_result$centers[, c("FamilyIncome", "TotalAmountSpent")], col = 1:num_clusters,
pch = 3, cex = 2)

# Add legend
legend("topright", legend = paste("Cluster", 1:num_clusters), col = 1:num_clusters, pch = 19, title
= "Clusters")
# View the result (first few rows)
head(customer_data)
```

Output:





```
> head(customer_data)
# A tibble: 6 × 5
  FamilyIncome TotalAmountSpent IsSubscriber SpendingCategory Cluster
         <dbl>            <dbl> <lgl>        <chr>            <fct>
1        41503             1100 FALSE        MediumSpenders   5
2        61532              833 TRUE         MediumSpenders   1
3        46359              989 FALSE        LowSpenders      3
4        65321             1454 TRUE         MediumSpenders   2
5        67619              983 FALSE        HighSpenders     2
6        31822             1390 TRUE         HighSpenders     4
>
```

Question 8:

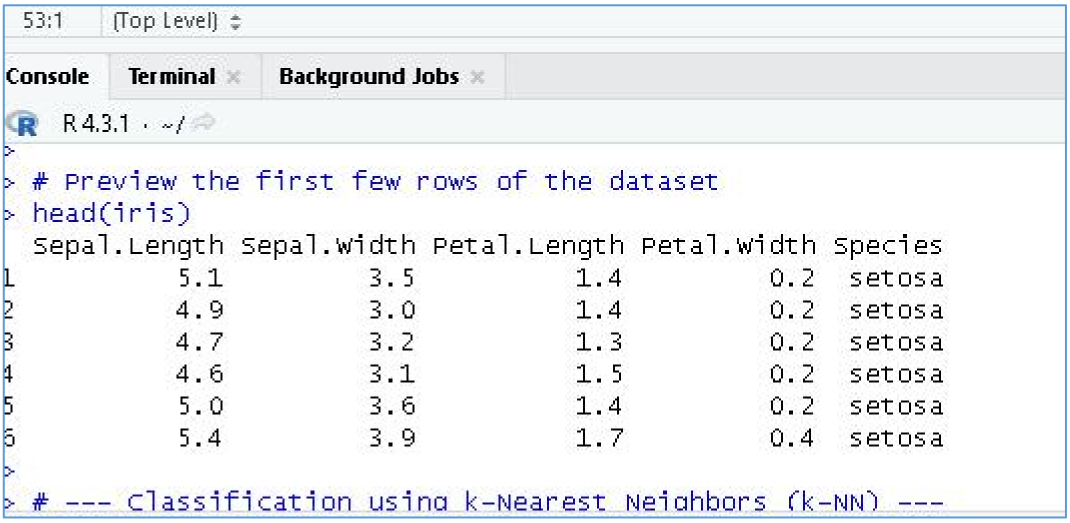
Perform classification and clustering for easily available datasets.

Program:

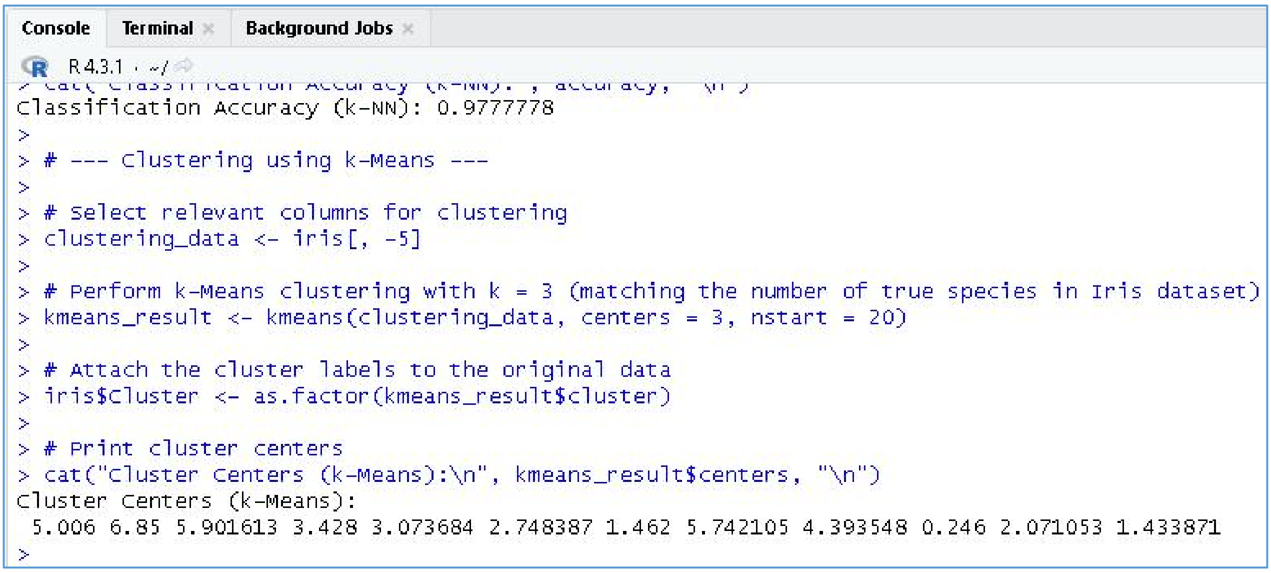
```
# Install class library for classification and clustering
install.packages("class")

# Install cluster library for clustering methods
install.packages("cluster")

# Load required libraries
library(class)
library(cluster)

# Load Iris dataset
data(iris)

# Preview the first few rows of the dataset
head(iris)

# --- Classification using k-Nearest Neighbors (k-NN) ---

# Split the dataset into training and testing sets
set.seed(123)
indices <- sample(1:nrow(iris), nrow(iris) * 0.7)  # 70% for training, 30% for testing
train_data <- iris[indices, ]
test_data <- iris[-indices, ]

# Perform k-NN classification
k <- 3
predicted_species <- knn(train = train_data[, -5], test = test_data[, -5], cl = train_data$Species, k
= k)

# Evaluate classification accuracy
accuracy <- sum(predicted_species == test_data$Species) / nrow(test_data)
cat("Classification Accuracy (k-NN):", accuracy, "\n")

# --- Clustering using k-Means ---

# Select relevant columns for clustering
clustering_data <- iris[, -5]

# Perform k-Means clustering with k = 3 (matching the number of true species in Iris dataset)
kmeans_result <- kmeans(clustering_data, centers = 3, nstart = 20)

# Attach the cluster labels to the original data
iris$Cluster <- as.factor(kmeans_result$cluster)
```

```
# Print cluster centers
cat("Cluster Centers (k-Means):\n", kmeans_result$centers, "\n")

# --- Visualize Results ---

# Plot k-NN classification results
plot(iris$Sepal.Length, iris$Sepal.Width, col = iris$Species, pch = 19, main = "k-NN Classification Results")
points(test_data$Sepal.Length, test_data$Sepal.Width, col = predicted_species, pch = 4)

# Plot k-Means clustering results
plot(iris$Sepal.Length, iris$Sepal.Width, col = iris$Cluster, pch = 19, main = "k-Means Clustering Results")
points(kmeans_result$centers[, c("Sepal.Length", "Sepal.Width")], col = 1:3, pch = 3, cex = 2)

# Display the plots
par(mfrow = c(1, 2))  # Set layout to side-by-side
```

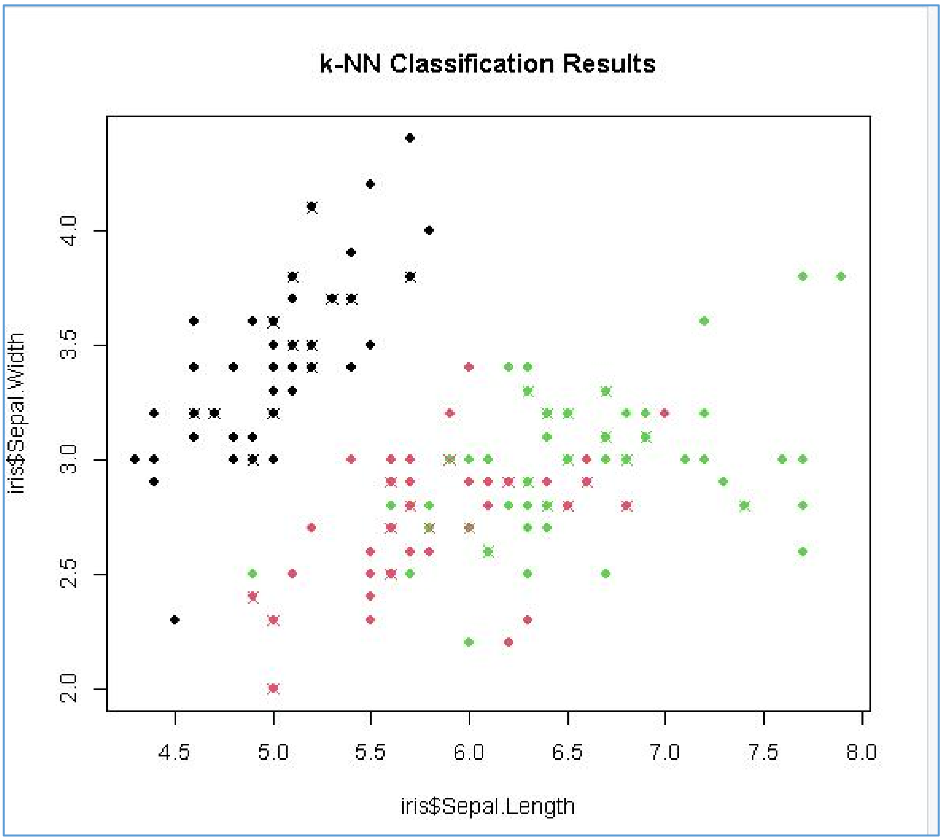
Output:

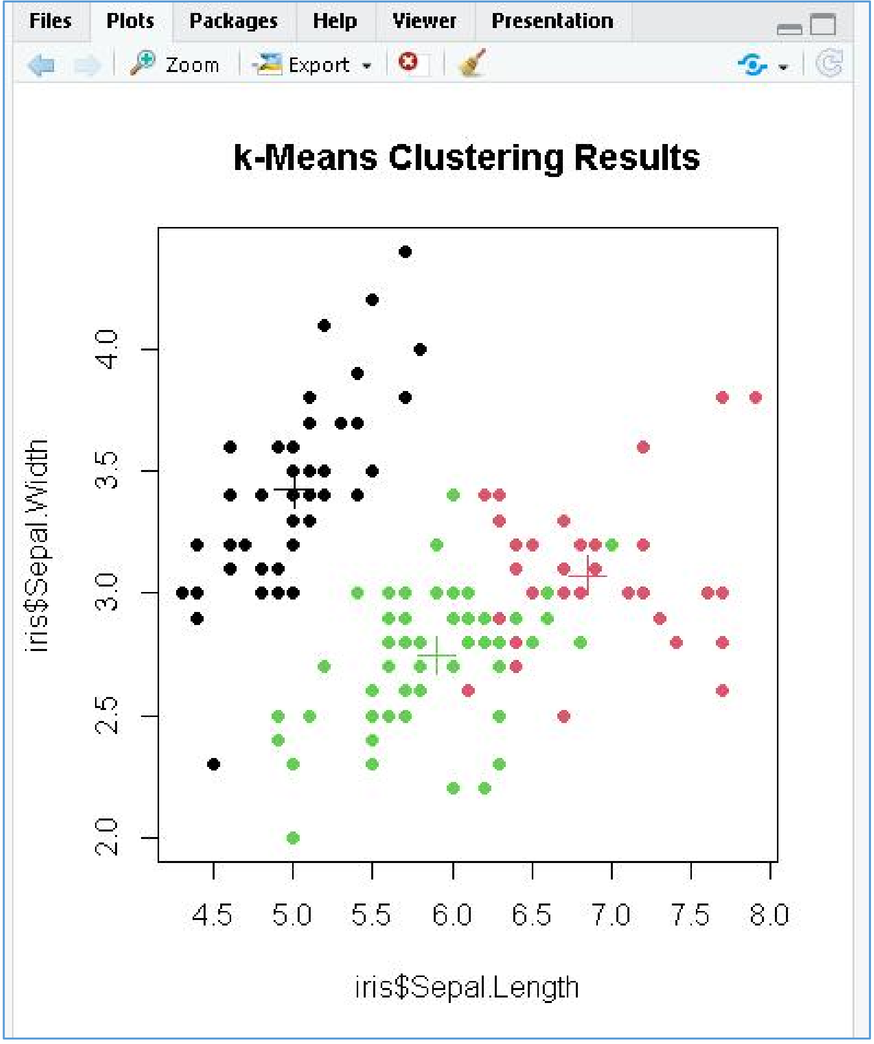


```
>
> # Preview the first few rows of the dataset
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
>
> # --- Classification using k-Nearest Neighbors (k-NN) ---
```



```
> cat("Classification Accuracy (k-NN):", accuracy, "\n")
Classification Accuracy (k-NN): 0.9777778
>
> # --- Clustering using k-Means ---
>
> # Select relevant columns for clustering
> clustering_data <- iris[, -5]
>
> # Perform k-Means clustering with k = 3 (matching the number of true species in Iris dataset)
> kmeans_result <- kmeans(clustering_data, centers = 3, nstart = 20)
>
> # Attach the cluster labels to the original data
> iris$Cluster <- as.factor(kmeans_result$cluster)
>
> # Print cluster centers
> cat("Cluster Centers (k-Means):\n", kmeans_result$centers, "\n")
Cluster Centers (k-Means):
 5.006 6.85 5.901613 3.428 3.073684 2.748387 1.462 5.742105 4.393548 0.246 2.071053 1.433871
>
```


k-NN Classification Results
iris$Sepal.Width
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
iris$Sepal.Length


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
k-Means Clustering Results
iris$Sepal.Width
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
iris$Sepal.Length

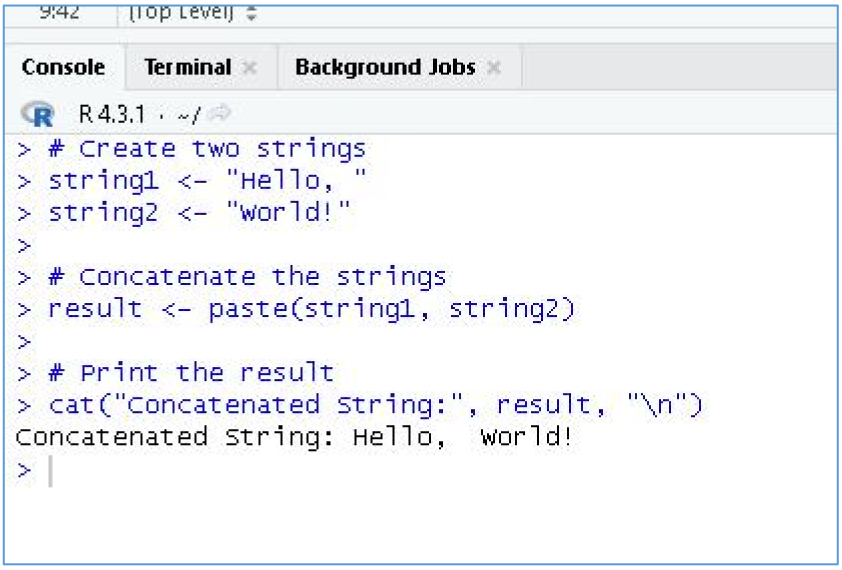
Question 9:

Create two strings and concatenate them.

Program:

```
# Create two strings
string1 <- "Hello, "
string2 <- "World!"

# Concatenate the strings
result <- paste(string1, string2)

# Print the result
cat("Concatenated String:", result, "\n")
```

Output:



```
> # Create two strings
> string1 <- "Hello, "
> string2 <- "World!"
>
> # Concatenate the strings
> result <- paste(string1, string2)
>
> # Print the result
> cat("Concatenated String:", result, "\n")
Concatenated String: Hello,  World!
> |
```

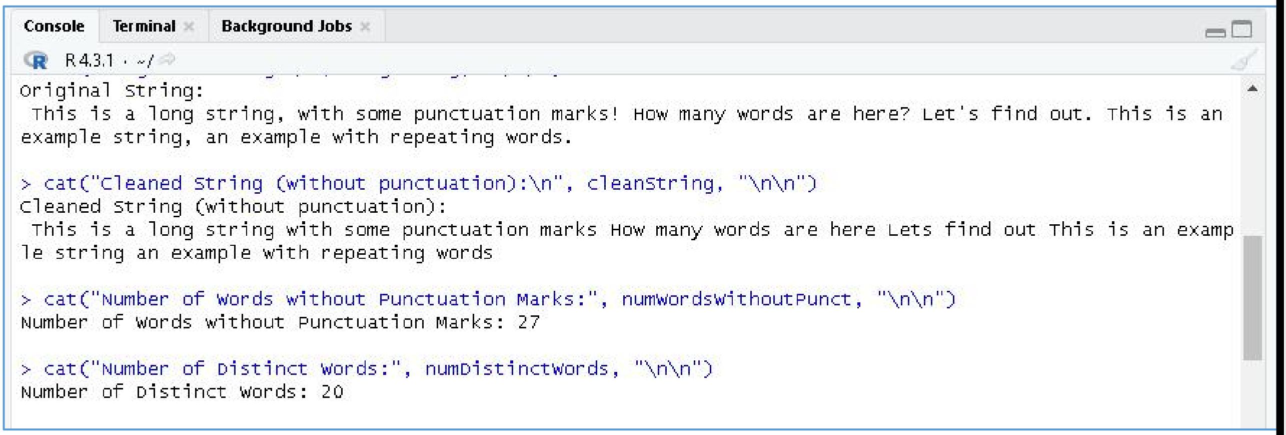
Question 10:

Create a long string of words separated by punctuation marks. Replace all the punctuation marks in the string using gsub("[[:punct:]]", "", stringName) function. Find the number of words in the string without punctuation marks. Find the number of distinct words and its count, if possible.

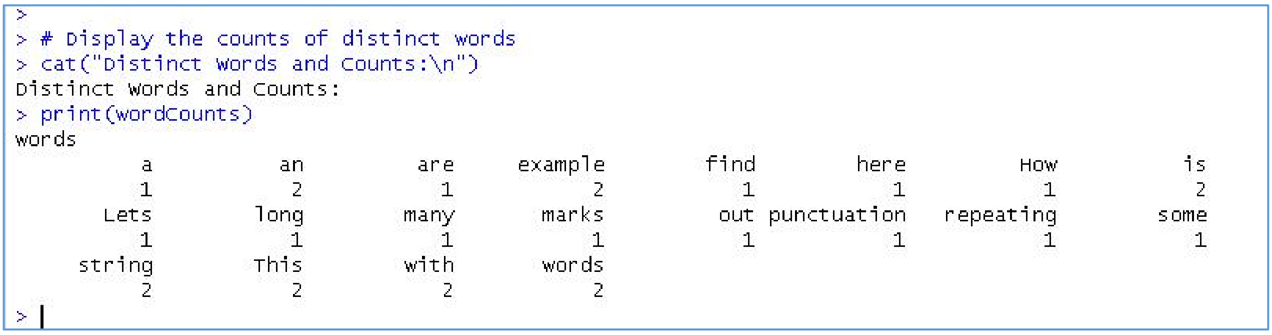
Program:

```
# Create a long string with punctuation marks
longString <- "This is a long string, with some punctuation marks! How many words are here? Let's find out. This is an example string, an example with repeating words."

# Replace punctuation marks with an empty string
cleanString <- gsub("[[:punct:]]", "", longString)

# Split the cleaned string into words
words <- strsplit(cleanString, "\\s+")[[1]]

# Find the number of words without punctuation marks
numWordsWithoutPunct <- length(words)

# Find the number of distinct words and their counts
wordCounts <- table(words)
numDistinctWords <- length(wordCounts)

# Print the results
cat("Original String:\n", longString, "\n\n")
cat("Cleaned String (without punctuation):\n", cleanString, "\n\n")
cat("Number of Words without Punctuation Marks:", numWordsWithoutPunct, "\n\n")
cat("Number of Distinct Words:", numDistinctWords, "\n\n")

# Display the counts of distinct words
cat("Distinct Words and Counts:\n")
print(wordCounts)
```

Output:



```
Original String:
 This is a long string, with some punctuation marks! How many words are here? Let's find out. This is an
example string, an example with repeating words.

> cat("Cleaned String (without punctuation):\n", cleanString, "\n\n")
Cleaned String (without punctuation):
 This is a long string with some punctuation marks How many words are here Lets find out This is an examp
le string an example with repeating words

> cat("Number of Words without Punctuation Marks:", numWordsWithoutPunct, "\n\n")
Number of Words without Punctuation Marks: 27

> cat("Number of Distinct Words:", numDistinctWords, "\n\n")
Number of Distinct Words: 20
```



```
>
> # Display the counts of distinct words
> cat("Distinct Words and Counts:\n")
Distinct Words and Counts:
> print(wordCounts)
words
          a          an         are     example        find        here         How          is
          1           2           1           2           1           1           1           2
       Lets        long        many       marks         out punctuation   repeating        some
          1           1           1           1           1           1           1           1
     string        This        with       words
          2           2           2           2
>
```

Question 11:

Store content in external files for the following types of data in R:

(i) Vectors (ii) Lists (iii) Arrays (iv) Data frames (v) Factors

Read those contents into R. Perform operations - sorting on vector data, finding the length of lists and adding data items in list, accessing different elements of array and comparing it to other values, accessing different components of data frames and factors.

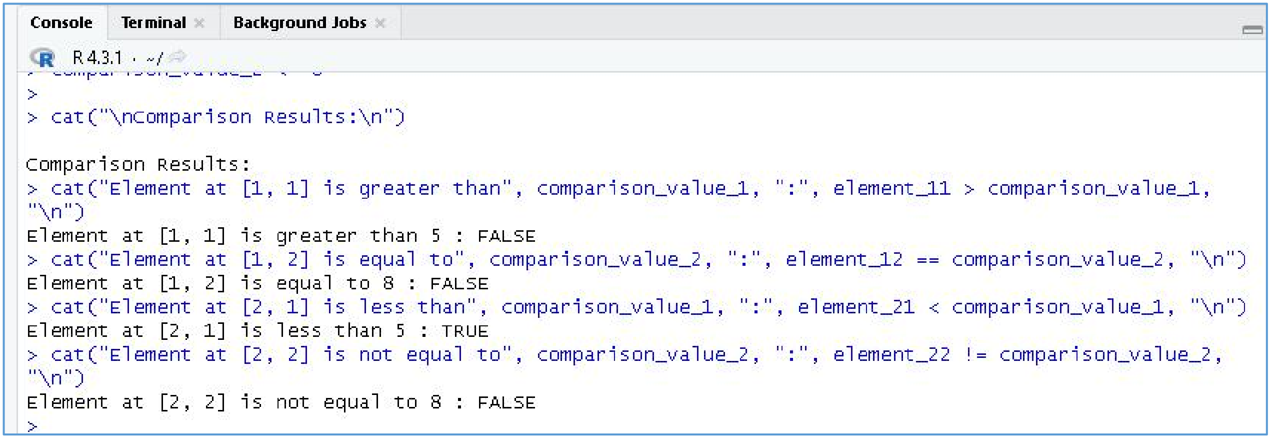
Program:

```
# i) Reading and operations on vectors
read_vector <- as.numeric(read.table("D:/IGNOULab/vector_data.txt")[, 1])
sorted_vector <- sort(read_vector)
cat("Original Vector:", read_vector, "\n")
cat("Sorted Vector:", sorted_vector, "\n\n")


# ii) Read the list from the text file
read_list <- read.table("D:/IGNOULab/list_data.txt", header = TRUE, sep = "\t", stringsAsFactors
= FALSE)
list_data <- as.list(read_list)
# Function to find the length of each list component
find_list_length <- function(my_list) {
  sapply(my_list, length)
}

# Find the length of each list component
lengths <- find_list_length(list_data)

# Print the lengths
cat("Length of Each List Component:\n")
print(lengths)

# Function to add data items to the list
add_data_to_list <- function(my_list, new_data) {
  for (i in seq_along(my_list)) {
    my_list[[i]] <- c(my_list[[i]], new_data[[i]])
  }
  return(my_list)
}

# Add new data items to the list
new_data <- list(
  name = "Eve",
  age = 28,
  grade = "A"
)

updated_list_data <- add_data_to_list(list_data, new_data)
```

```
# Print the updated list
cat("Updated List Data:\n")
print(updated_list_data)

# iii) Read the 2x2 array from the text file
read_array <- matrix(scan("D:/IGNOULab/array_data.txt"), nrow = 2, byrow = TRUE)

# Display the entire array
cat("Original Array:\n")
print(read_array)

# Access elements in the 2x2 array
element_11 <- read_array[1, 1]
element_12 <- read_array[1, 2]
element_21 <- read_array[2, 1]
element_22 <- read_array[2, 2]

# Compare elements to other values
comparison_value_1 <- 5
comparison_value_2 <- 8

cat("\nComparison Results:\n")
cat("Element at [1, 1] is greater than", comparison_value_1, ":", element_11 >
comparison_value_1, "\n")
cat("Element at [1, 2] is equal to", comparison_value_2, ":", element_12 == comparison_value_2,
"\n")
cat("Element at [2, 1] is less than", comparison_value_1, ":", element_21 < comparison_value_1,
"\n")
cat("Element at [2, 2] is not equal to", comparison_value_2, ":", element_22 !=
comparison_value_2, "\n")


# iv) Read data from the text file into a data frame
data_frame <- read.table("D:/IGNOULab/students_data.txt", header = TRUE)

# Display the entire data frame
cat("Original Data Frame:\n")
print(data_frame)

# Access different components of the data frame
names_column <- data_frame$Name
ages_column <- data_frame$Age
grades_column <- data_frame$Grade

# Display individual columns
cat("\nIndividual Columns:\n")
cat("Names Column:\n", names_column, "\n")
cat("Ages Column:\n", ages_column, "\n")
cat("Grades Column:\n", grades_column, "\n")

# Access and display specific values
element_11 <- data_frame[1, 1]  # Name of the first person
element_23 <- data_frame[2, 3]  # Grade of the second person

cat("\nSpecific Values:\n")
cat("Element at [1, 1]:", element_11, "\n")
cat("Element at [2, 3]:", element_23, "\n")

# v) Convert the 'Grade' column to a factor
data_frame$Grade <- as.factor(data_frame$Grade)

# Display the levels of the 'Grade' factor
cat("\nLevels of the 'Grade' Factor:\n")
print(levels(data_frame$Grade))
```

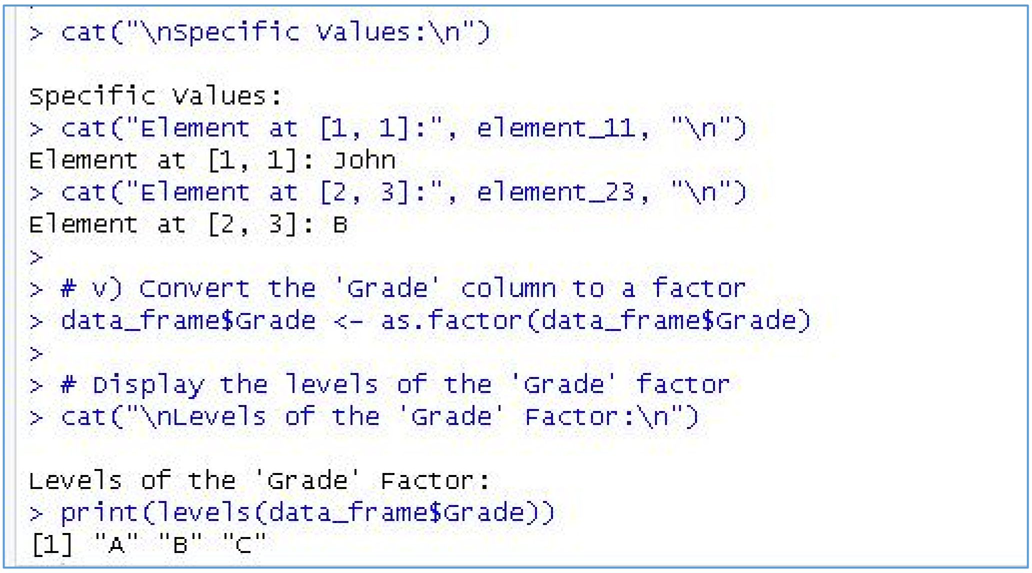
Output:

Array:



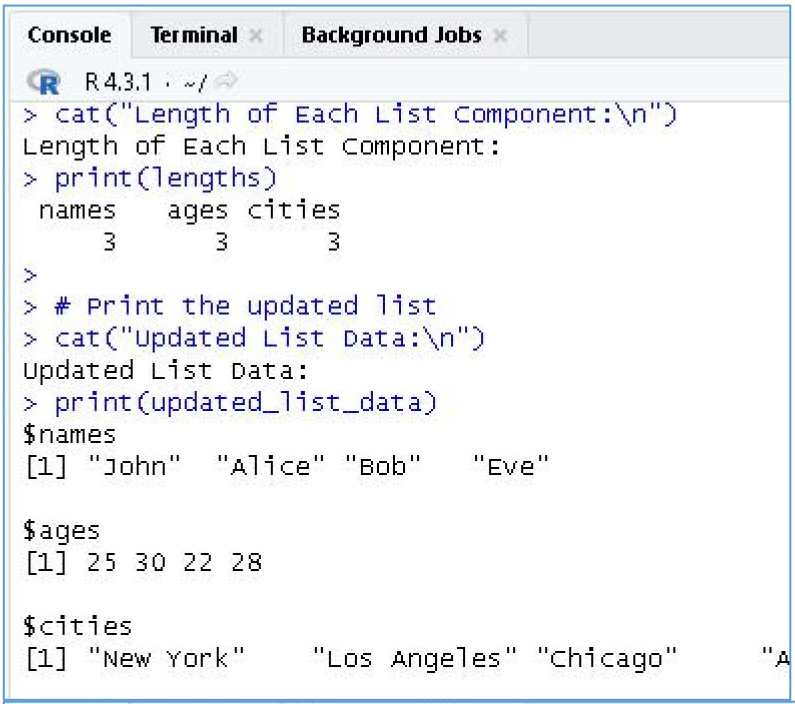
```
> comparison_value_2 <- 8
>
> cat("\nComparison Results:\n")

Comparison Results:
> cat("Element at [1, 1] is greater than", comparison_value_1, ":", element_11 > comparison_value_1,
"\n")
Element at [1, 1] is greater than 5 : FALSE
> cat("Element at [1, 2] is equal to", comparison_value_2, ":", element_12 == comparison_value_2, "\n")
Element at [1, 2] is equal to 8 : FALSE
> cat("Element at [2, 1] is less than", comparison_value_1, ":", element_21 < comparison_value_1, "\n")
Element at [2, 1] is less than 5 : TRUE
> cat("Element at [2, 2] is not equal to", comparison_value_2, ":", element_22 != comparison_value_2,
"\n")
Element at [2, 2] is not equal to 8 : FALSE
>
```

Data Frame and Factor:

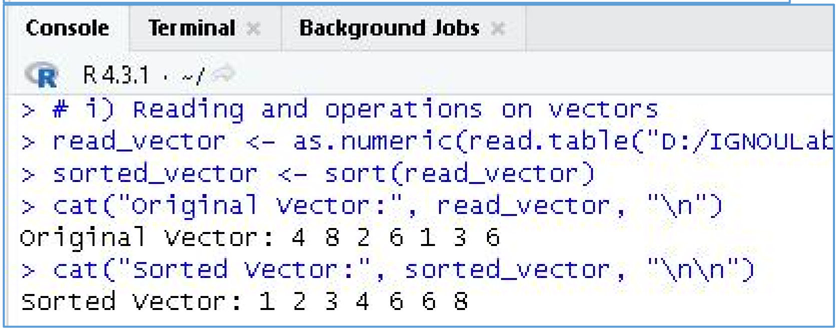


```
> cat("\nSpecific Values:\n")

Specific Values:
> cat("Element at [1, 1]:", element_11, "\n")
Element at [1, 1]: John
> cat("Element at [2, 3]:", element_23, "\n")
Element at [2, 3]: B
>
> # v) Convert the 'Grade' column to a factor
> data_frame$Grade <- as.factor(data_frame$Grade)
>
> # Display the levels of the 'Grade' factor
> cat("\nLevels of the 'Grade' Factor:\n")

Levels of the 'Grade' Factor:
> print(levels(data_frame$Grade))
[1] "A" "B" "C"
```

List and Vector:



```
> cat("Length of Each List Component:\n")
Length of Each List Component:
> print(lengths)
 names   ages cities 
     3      3      3 
>
> # Print the updated list
> cat("Updated List Data:\n")
Updated List Data:
> print(updated_list_data)
$names
[1] "John"  "Alice" "Bob"   "Eve"  

$ages
[1] 25 30 22 28

$cities
[1] "New York"    "Los Angeles" "Chicago"     "A
```



```
> # i) Reading and operations on vectors
> read_vector <- as.numeric(read.table("D:/IGNOULab
> sorted_vector <- sort(read_vector)
> cat("Original Vector:", read_vector, "\n")
Original Vector: 4 8 2 6 1 3 6 
> cat("Sorted Vector:", sorted_vector, "\n\n")
Sorted Vector: 1 2 3 4 6 6 8 
```

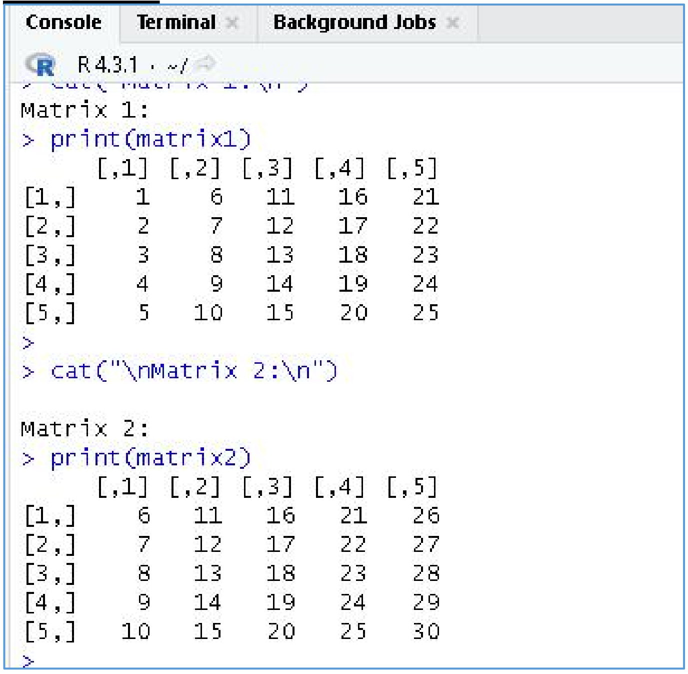
Question 12:

Create two matrices of 5 * 5 : size using R, add, subtract and multiply these two matrices.

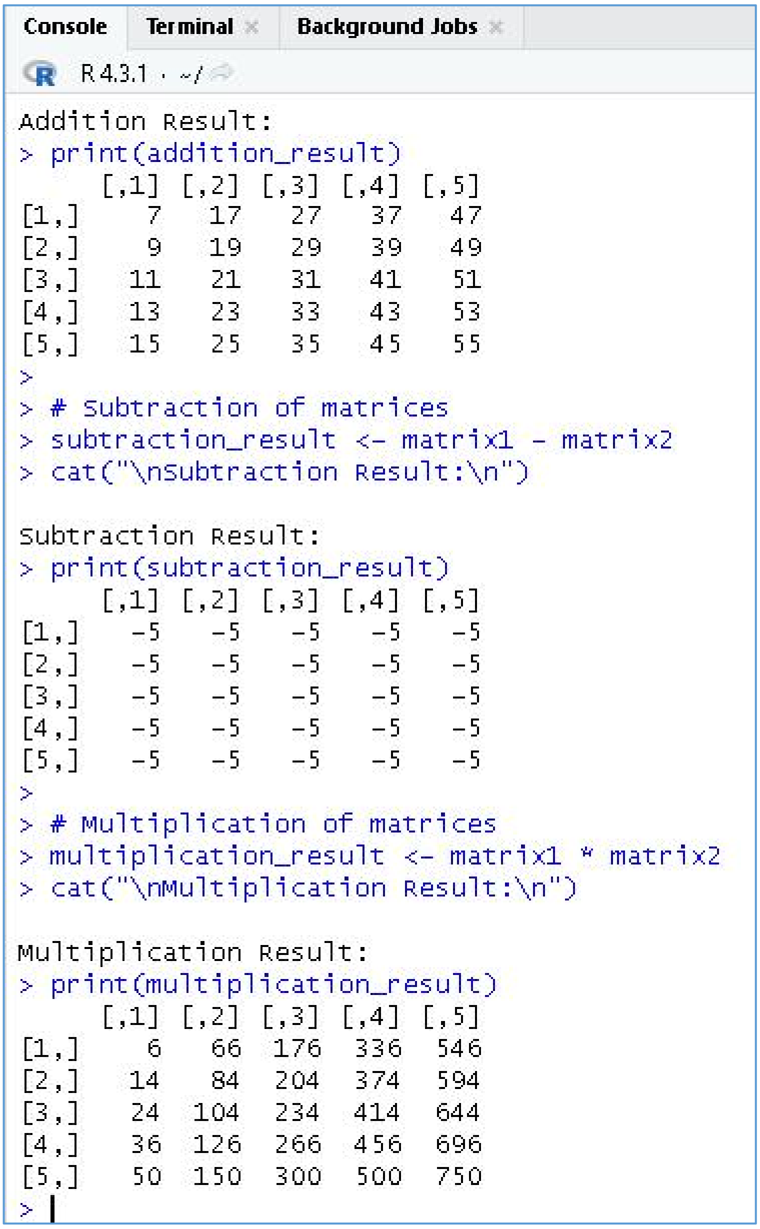
Program:

```
# Create two 5x5 matrices
matrix1 <- matrix(1:25, nrow = 5)
matrix2 <- matrix(6:30, nrow = 5)

# Display the original matrices
cat("Matrix 1:\n")
print(matrix1)

cat("\nMatrix 2:\n")
print(matrix2)

# Addition of matrices
addition_result <- matrix1 + matrix2
cat("\nAddition Result:\n")
print(addition_result)

# Subtraction of matrices
subtraction_result <- matrix1 - matrix2
cat("\nSubtraction Result:\n")
print(subtraction_result)

# Multiplication of matrices
multiplication_result <- matrix1 * matrix2
cat("\nMultiplication Result:\n")
print(multiplication_result)
```

Output:



```
Matrix 1:
> print(matrix1)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11   16   21
[2,]    2    7   12   17   22
[3,]    3    8   13   18   23
[4,]    4    9   14   19   24
[5,]    5   10   15   20   25
>
> cat("\nMatrix 2:\n")

Matrix 2:
> print(matrix2)
     [,1] [,2] [,3] [,4] [,5]
[1,]    6   11   16   21   26
[2,]    7   12   17   22   27
[3,]    8   13   18   23   28
[4,]    9   14   19   24   29
[5,]   10   15   20   25   30
>
```



```
Addition Result:
> print(addition_result)
     [,1] [,2] [,3] [,4] [,5]
[1,]    7   17   27   37   47
[2,]    9   19   29   39   49
[3,]   11   21   31   41   51
[4,]   13   23   33   43   53
[5,]   15   25   35   45   55
>
> # Subtraction of matrices
> subtraction_result <- matrix1 - matrix2
> cat("\nSubtraction Result:\n")

Subtraction Result:
> print(subtraction_result)
     [,1] [,2] [,3] [,4] [,5]
[1,]   -5   -5   -5   -5   -5
[2,]   -5   -5   -5   -5   -5
[3,]   -5   -5   -5   -5   -5
[4,]   -5   -5   -5   -5   -5
[5,]   -5   -5   -5   -5   -5
>
> # Multiplication of matrices
> multiplication_result <- matrix1 * matrix2
> cat("\nMultiplication Result:\n")

Multiplication Result:
> print(multiplication_result)
     [,1] [,2] [,3] [,4] [,5]
[1,]    6   66  176  336  546
[2,]   14   84  204  374  594
[3,]   24  104  234  414  644
[4,]   36  126  266  456  696
[5,]   50  150  300  500  750
> 
```

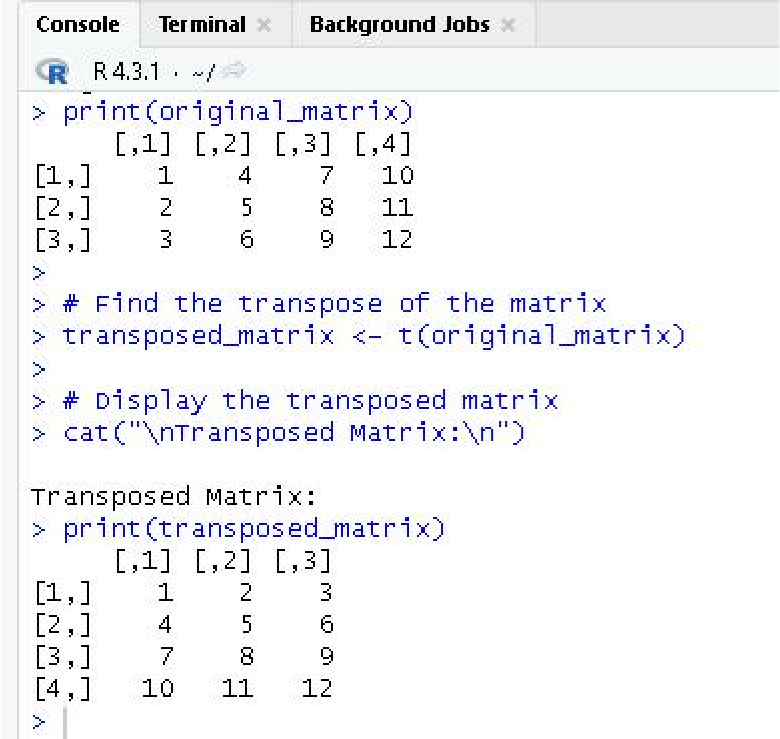
Question 13:

Find the transpose of a matrix.

Program:

```
# Create a sample 3x4 matrix
original_matrix <- matrix(1:12, nrow = 3, ncol = 4)

# Display the original matrix
cat("Original Matrix:\n")
print(original_matrix)

# Find the transpose of the matrix
transposed_matrix <- t(original_matrix)

# Display the transposed matrix
cat("\nTransposed Matrix:\n")
print(transposed_matrix)
```

Output:



```
> print(original_matrix)
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> 
> # Find the transpose of the matrix
> transposed_matrix <- t(original_matrix)
> 
> # Display the transposed matrix
> cat("\nTransposed Matrix:\n")

Transposed Matrix:
> print(transposed_matrix)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
> 
```

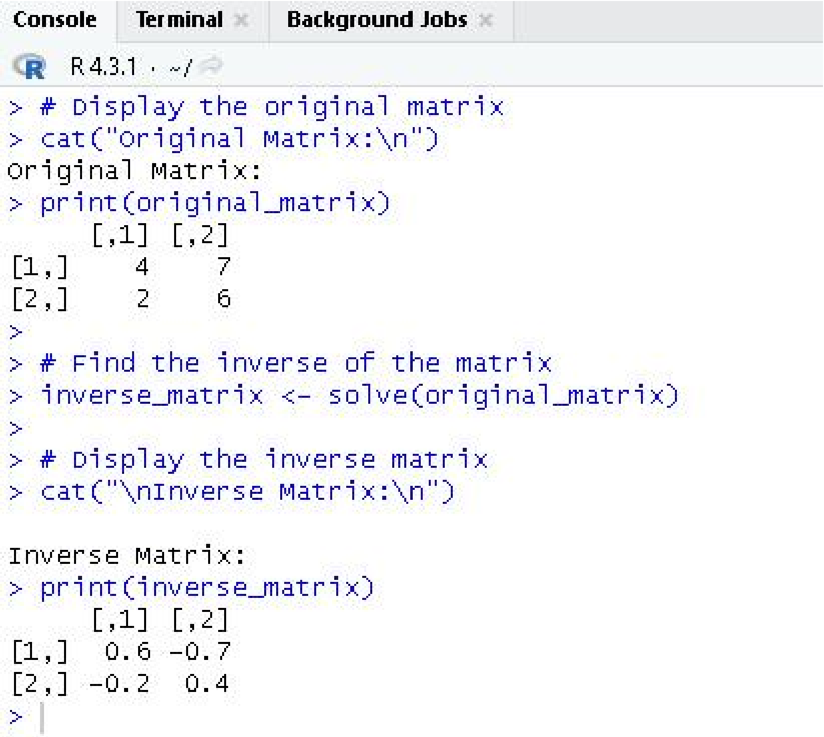
Question 14:

Find the inverse of a matrix.

Program:

```
# Create a sample 2x2 matrix
original_matrix <- matrix(c(4, 7, 2, 6), nrow = 2, byrow = TRUE)

# Display the original matrix
cat("Original Matrix:\n")
print(original_matrix)

# Find the inverse of the matrix
inverse_matrix <- solve(original_matrix)

# Display the inverse matrix
cat("\nInverse Matrix:\n")
print(inverse_matrix)
```

Output:



```
> # Display the original matrix
> cat("Original Matrix:\n")
Original Matrix:
> print(original_matrix)
     [,1] [,2]
[1,]    4    7
[2,]    2    6
>
> # Find the inverse of the matrix
> inverse_matrix <- solve(original_matrix)
>
> # Display the inverse matrix
> cat("\nInverse Matrix:\n")

Inverse Matrix:
> print(inverse_matrix)
     [,1] [,2]
[1,]  0.6 -0.7
[2,] -0.2  0.4
> 
```

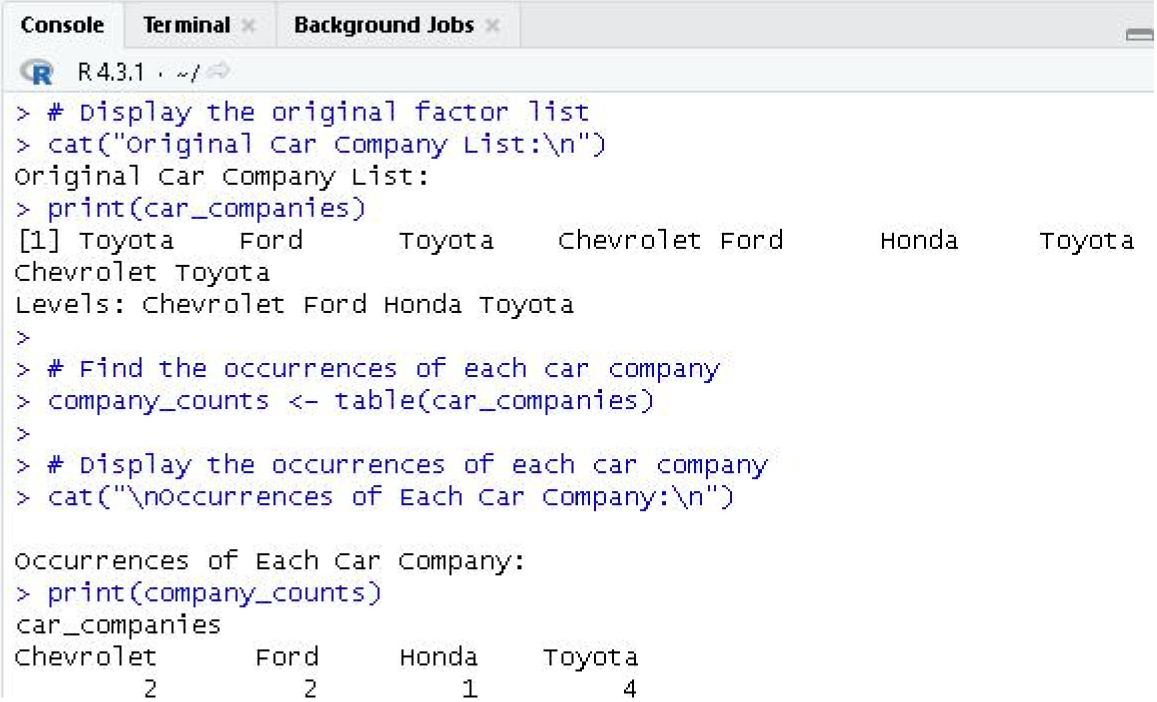
Question 15:

Create a list of factors. Find the occurrences of each factor in the list.

Program:

```
# Create a list of car company names as factors
car_companies <- as.factor(c("Toyota", "Ford", "Toyota", "Chevrolet", "Ford", "Honda", "Toyota",
"Chevrolet", "Toyota"))

# Display the original factor list
cat("Original Car Company List:\n")
print(car_companies)

# Find the occurrences of each car company
company_counts <- table(car_companies)

# Display the occurrences of each car company
cat("\nOccurrences of Each Car Company:\n")
print(company_counts)
```

Output:



```
> # Display the original factor list
> cat("Original Car Company List:\n")
Original Car Company List:
> print(car_companies)
[1] Toyota    Ford      Toyota    Chevrolet Ford      Honda     Toyota
Chevrolet Toyota
Levels: Chevrolet Ford Honda Toyota
>
> # Find the occurrences of each car company
> company_counts <- table(car_companies)
>
> # Display the occurrences of each car company
> cat("\nOccurrences of Each Car Company:\n")

Occurrences of Each Car Company:
> print(company_counts)
car_companies
Chevrolet      Ford     Honda    Toyota
        2         2         1         4
```