

Azure Cloud Computing Project: Implementation and Hands-on Experience

INTRODUCTION

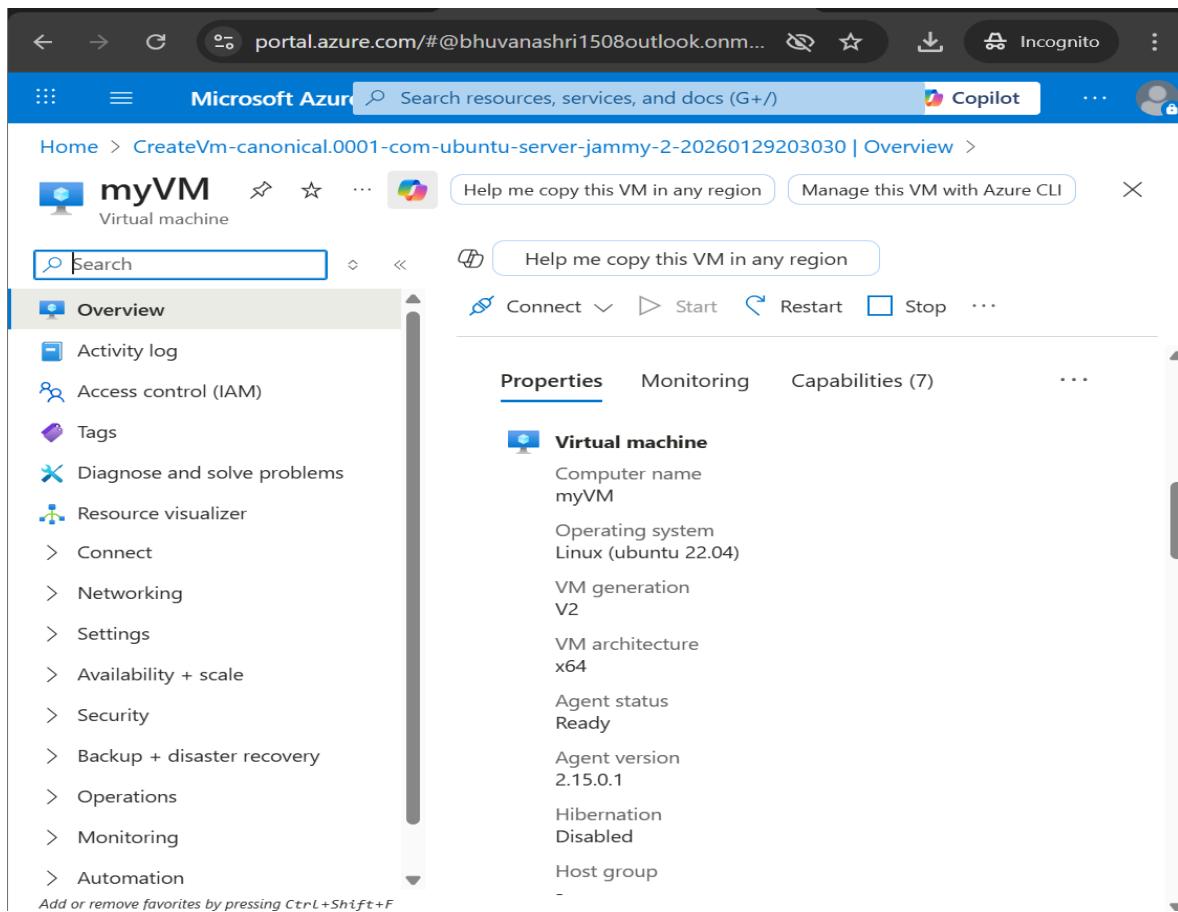
This project demonstrates an end-to-end cloud-based data analytics solution using Microsoft Azure and Power BI. The objective is to design, deploy, and analyze structured business data by integrating cloud database services with data visualization tools. The AdventureWorksLT dataset is used to represent a real-world sales and business scenario.

An Azure SQL Database is created to securely store transactional data, which is explored and validated using SQL queries. The database is then connected to Power BI Desktop to create calculated measures and interactive visualizations that analyze sales performance, product distribution, order volume, and yearly trends. The project showcases the practical application of cloud computing, SQL, and business intelligence for data-driven decision-making.

EXECUTION

Step 1: Azure Virtual Machine – Overview Verification

- Navigated to the **Virtual Machine Overview** page in the Azure Portal.
- Verified that the Ubuntu VM **myVM** was created successfully.
- Confirmed the operating system as **Linux (Ubuntu 22.04 LTS)**.
- Checked VM status and ensured the **Azure agent status is Ready**.
- Verified that VM management options such as **Connect, Start, Restart, and Stop** are available.



Step 2: Viewing the Virtual Machine in Azure Portal

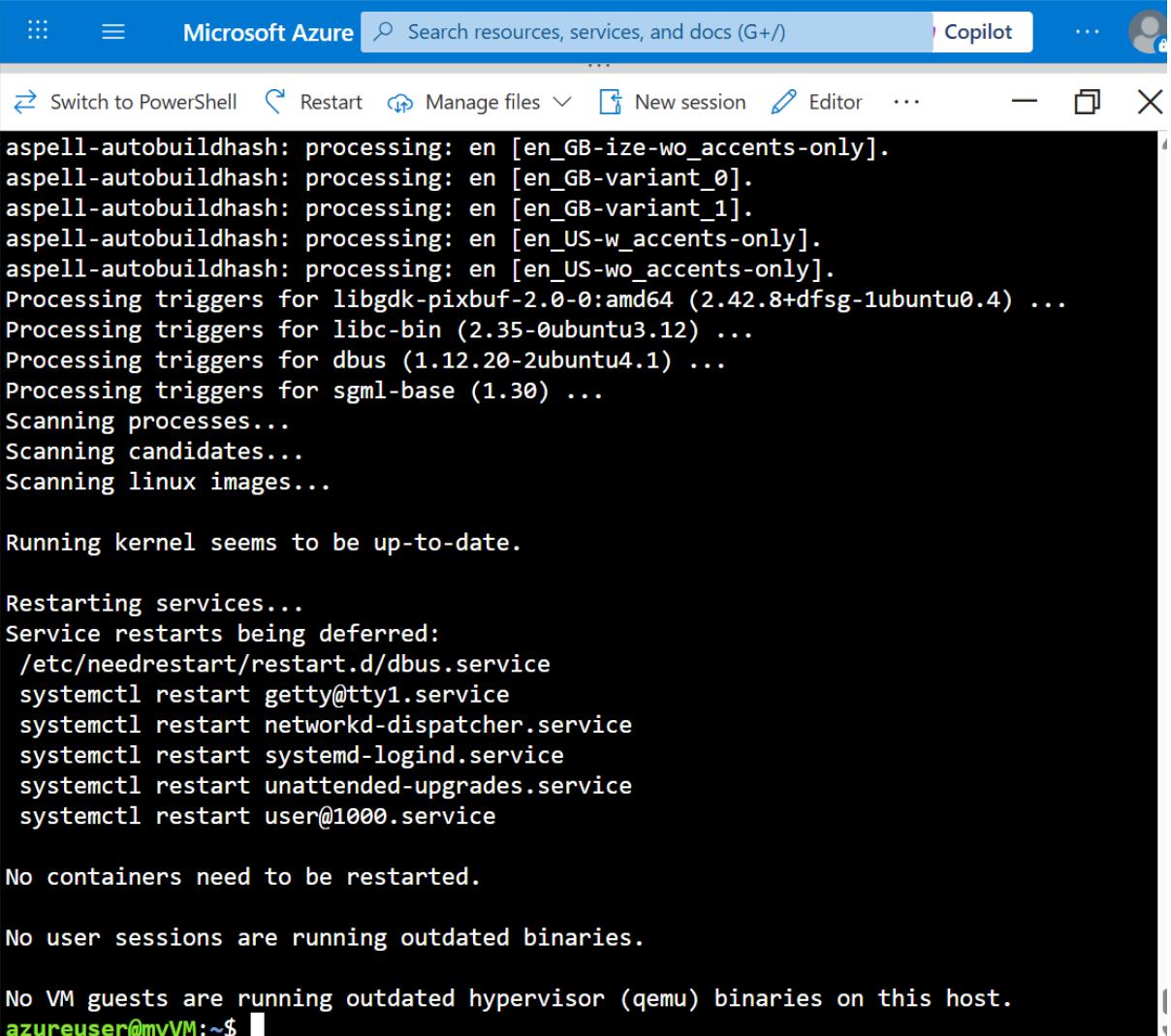
- Navigated to **Compute Infrastructure → Virtual Machines** in the Azure Portal.
- Verified that the virtual machine **myVM** is listed under the active subscription.
- Confirmed the VM is associated with the resource group **myVM_group**.
- Verified the **location** of the VM as **Central India**.
- Checked the VM **status** as **Running**, confirming successful deployment.
- Confirmed the **operating system** as **Linux** and the **VM size** as **Standard_D4s_v3**.

The screenshot shows the Microsoft Azure Compute Infrastructure Virtual Machines page. The left sidebar is collapsed, and the main area displays a table of virtual machines. One row is selected, showing details for a VM named 'myVM'. The table columns include Name, Subscription, Resource Group, Location, Status, Operating system, and Size. The 'myVM' row shows 'Name' as 'myVM', 'Subscription' as 'Azure subscript...', 'Resource Group' as 'myVM_group', 'Location' as 'Central India', 'Status' as 'Running', 'Operating system' as 'Linux', and 'Size' as 'Standard_D4s_v3'. A filter bar at the top allows for searching by various criteria like Name, Subscription, Resource Group, Location, Status, Operating system, and Size. A message at the top indicates a new version of the Browse experience is available.

Name	Subscription	Resource Group	Location	Status	Operating system	Size
myVM	Azure subscript...	myVM_group	Central India	Running	Linux	Standard_D4s_v3

Step 3: Resetting Login Credentials for the Ubuntu Virtual Machine

- Opened the **myVM** → **Reset password** option from the Azure Portal.
- Selected the **Reset password** mode using the **VMAccessForLinux** extension.
- Reset the credentials for the existing Linux user to ensure successful login access.
- Applied the changes by clicking **Update**, allowing secure authentication to the VM.
- This step ensures proper user access for connecting to the Ubuntu VM via SSH or RDP.

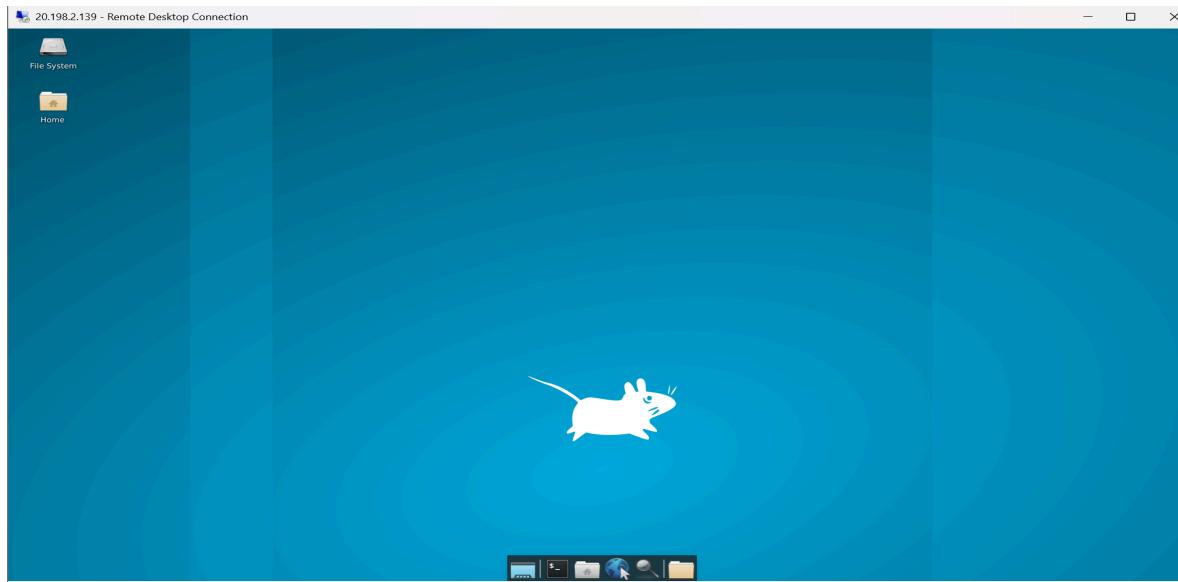
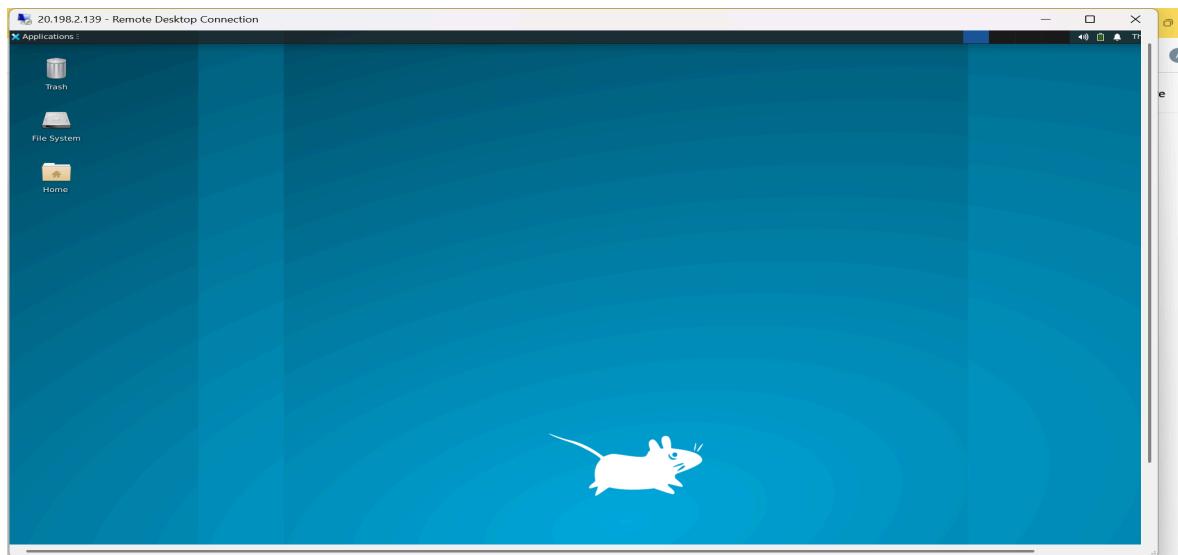


The screenshot shows a Microsoft Azure Cloud Shell interface. The terminal window displays the following output:

```
aspell-autobuildhash: processing: en [en_GB-ize-wo_accents-only].  
aspell-autobuildhash: processing: en [en_GB-variant_0].  
aspell-autobuildhash: processing: en [en_GB-variant_1].  
aspell-autobuildhash: processing: en [en_US-w_accents-only].  
aspell-autobuildhash: processing: en [en_US-wo_accents-only].  
Processing triggers for libgdk-pixbuf-2.0-0:amd64 (2.42.8+dfsg-1ubuntu0.4) ...  
Processing triggers for libc-bin (2.35-0ubuntu3.12) ...  
Processing triggers for dbus (1.12.20-2ubuntu4.1) ...  
Processing triggers for sgml-base (1.30) ...  
Scanning processes...  
Scanning candidates...  
Scanning linux images...  
  
Running kernel seems to be up-to-date.  
  
Restarting services...  
Service restarts being deferred:  
/etc/needrestart/restart.d/dbus.service  
systemctl restart getty@tty1.service  
systemctl restart networkd-dispatcher.service  
systemctl restart systemd-logind.service  
systemctl restart unattended-upgrades.service  
systemctl restart user@1000.service  
  
No containers need to be restarted.  
  
No user sessions are running outdated binaries.  
  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
azureuser@myVM:~$
```

Step 4: Connecting to the Ubuntu Virtual Machine Using Remote Desktop (RDP)

- Established a **Remote Desktop Connection (RDP)** to the Ubuntu Virtual Machine using its public IP address.
- Successfully logged in using the configured Linux user credentials.
- Verified that the **Ubuntu GUI desktop environment** is displayed.
- Confirmed that the graphical interface is accessible, indicating successful installation of **GUI and XRDP**.
- This step validates that the Ubuntu VM can be accessed remotely using a graphical desktop interface.



TASK 2: Azure Storage Account

Step 1: Creating an Azure Storage Account

- Initiated the creation of a new **Azure Storage Account** from the Azure Portal.
- Verified that the deployment completed successfully.
- Confirmed the storage account name as **newsorage_176977639852**.
- Associated the storage account with the resource group **myVM_group**.
- Verified the storage account location as **Central India**.

- Ensured the provisioning state shows **Succeeded**.

The screenshot shows the Microsoft Azure portal's deployment overview page. The deployment name is `newsorage_1769776398582`. The status message says "Your deployment is complete". Deployment details include the name, subscription, and resource group. A sidebar on the right provides links to cost management, Microsoft Defender for Cloud, and free tutorials.

Step 2: Accessing the Storage Account Overview

- Navigated to the **Overview** section of the newly created storage account.
- Verified the **account kind** as **StorageV2 (General-purpose v2)**.
- Confirmed the **replication type** as **Locally Redundant Storage (LRS)**.
- Checked the **performance tier** as **Standard**.
- Ensured the storage account is ready for further configuration.

The screenshot shows the Microsoft Azure portal's storage account overview page for `newsorage_1769776398582`. Key properties listed include:

- Resource group: `myVM_group`
- Location: `centralindia`
- Subscription: `Azure subscription 1`
- Subscription ID: `559191e0-5a76-48c4-b88a-258e03c94f16`
- Disk state: `Available`
- Tags: `(edit) Add tags`
- Properties: `Performance: Standard, Replication: Locally-redundant storage (LRS), Account kind: StorageV2 (general purpose v2), Provisioning state: Succeeded, Created: 1/30/2026, 6:03:41 PM`

Step 3: Configuring Storage Account Settings for Public Blob Access

- Opened the **Configuration** section of the storage account **newsorage_1769776398582**.
- Verified that **Secure transfer required** is enabled.
- Enabled **Allow Blob anonymous access** to make blobs publicly accessible.
- Confirmed that **storage account key access** is enabled.
- Reviewed additional security and access settings to ensure compatibility with static website hosting.
- Saved the configuration changes.

The screenshot shows the Microsoft Azure Storage Configuration page for the storage account **newsorage_1769776398582**. The page displays several configuration options:

- Secure transfer required**: Enabled (radio button selected)
- Managed Identity for SMB**: Disabled (radio button selected)
- Allow Blob anonymous access**: Enabled (radio button selected)
- Allow storage account key access**: Enabled (radio button selected)
- Allow upper limit for shared access signature (SAS) expiry interval**: Disabled (radio button selected)
- Default to Microsoft Entra authorization in the Azure portal**: Disabled (radio button selected)
- Minimum TLS version**: Version 1.0

Step 4: Creating a Blob Container

- Navigated to **Data storage → Containers** within the storage account **newsorage_1769776398582**.
- Created a new blob container named **images**.
- Set the **anonymous access level** to **Blob**, allowing public read access to blobs.
- Verified that the container status is **Available**.
- Confirmed the container creation time and successful setup.

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes links for 'Microsoft Azure', 'Upgrade', 'Search resources, services, and docs (G+)', 'Copilot', and user profile information. Below the navigation is a breadcrumb trail: 'Home > newsorage_1769776398582 | Overview > newsorage'. The main content area is titled 'newsorage | Containers' and describes it as a 'Storage account'. On the left, a sidebar menu lists various options: Overview, Access Control (IAM), Data storage (Containers is selected), Security + networking, Settings (Configuration, Endpoints), Help (Connectivity check, Support + Troubleshooting), and a search bar for containers by prefix. The right side displays a table with one item: 'Name' (images), 'Last modified' (1/30/2026, 6:20:33 PM), 'Anonymous access level' (Blob), and 'Lease state' (Available). There are also buttons for 'Add container', 'Upload', 'Refresh', 'Delete', 'Change access level', 'Restore containers', and 'Edit columns'.

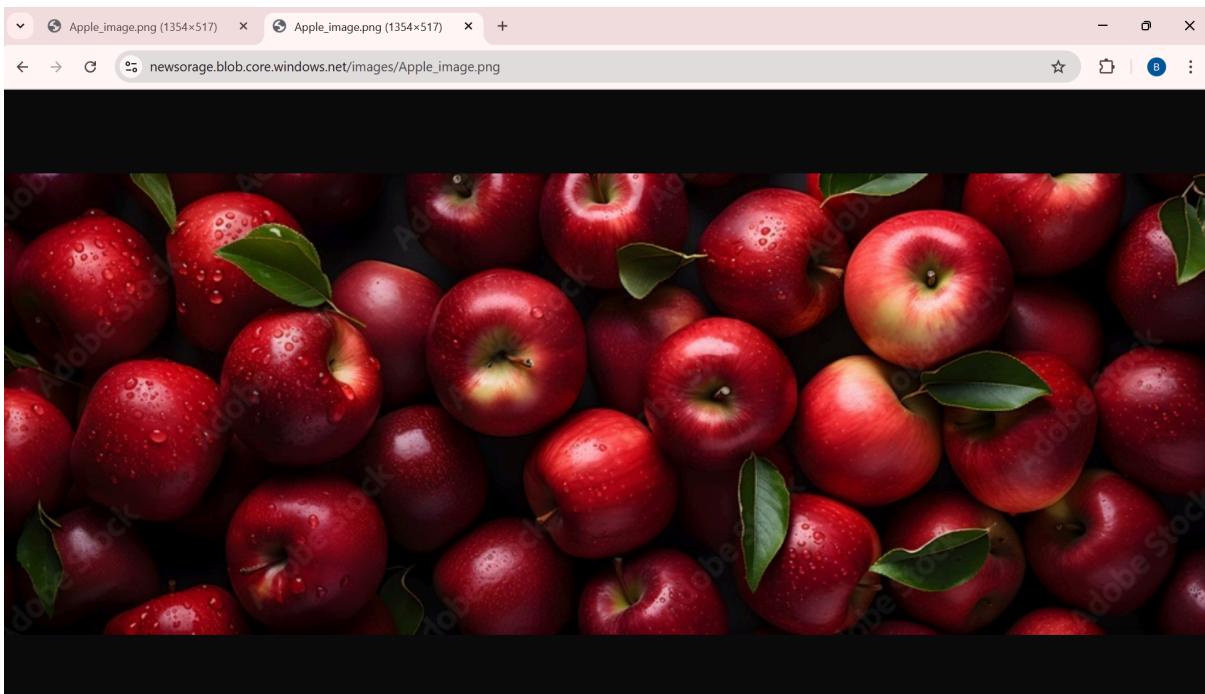
Name	Last modified	Anonymous access level	Lease state
images	1/30/2026, 6:20:33 PM	Blob	Available

Step 5: Uploading an Image to Blob Storage and Verifying Public Access

- Opened the **images** blob container within the storage account.
- Uploaded an image file named **Apple_image.png** to the container.
- Verified the blob type as **Block blob** with the access tier set to **Hot**.
- Confirmed that the image was successfully uploaded and is listed in the container.
- Accessed the image using the **blob URL** to verify that it is publicly accessible.
- Successfully viewed the image in a web browser, confirming public access configuration.

The screenshot shows the Microsoft Azure Storage Container Overview page for the 'images' container. At the top, there's a search bar and a toolbar with options like 'Add Directory', 'Upload', 'Change access level', 'Refresh', 'Delete', 'Copy', 'Paste', 'Rename', 'Acquire lease', and more. Below the toolbar, there's a sidebar with links for 'Overview', 'Diagnose and solve problems', 'Access Control (IAM)', and 'Settings'. The main area displays a table with one item: 'Apple_image.png'. The table columns are 'Name', 'Last modified', 'Access tier', 'Blob type', 'Size', and 'Lease state'. The blob details are: Name - Apple_image.png, Last modified - 1/30/2026, 6:28:29 PM, Access tier - Hot (Inferred), Blob type - Block blob, Size - 1011.37 KiB, Lease state - Available.

Image url: https://newsorage.blob.core.windows.net/images/Apple_image.png



Step 6: Enabling Static Website Hosting

- Navigated to Data management → Static website in the storage account.

- Enabled the **Static website** option.
- Set the **Index document name** as **index.html**.
- Set the **Error document path** as **404.html**.
- Noted the **primary endpoint URL** generated for the static website.
- Confirmed that the special **\$web container** was automatically created.

The screenshot shows the Microsoft Azure portal interface for managing a storage account named 'newsorage'. The 'Static website' tab is active in the left sidebar. Key configuration details shown include:

- Primary endpoint:** https://newsorage.z29.web.core.windows.net/
- Index document name:** index.html
- Error document path:** 404.html
- A note states: "Enabling static websites on the blob service allows you to host static content. Webpages may include static content and client-side scripts. Server-side scripting is not supported. As data is replicated asynchronously from primary to secondary regions, files at the secondary endpoint may not be immediately available or in sync with files at the primary endpoint." It also mentions "Learn more" and "Azure Front Door".

HelloWorld website URL:

<https://newsorage.z29.web.core.windows.net/>

Step 7: Hosting and Verifying the Hello World Webpage

- Opened the **\$web** container created for static website hosting.
- Uploaded the **index.html** file containing the Hello World content.
- Verified that the file is stored as a **block blob**.

- Accessed the static website using the provided endpoint URL.
- Confirmed that the Hello World webpage loads successfully in the browser.

The screenshot shows the Microsoft Azure Storage Container Overview page for a container named '\$web'. The page includes a search bar, navigation buttons, and a toolbar with options like Add Directory, Upload, Change access level, Refresh, Delete, Copy, Paste, Rename, and Acquire lease. A sidebar on the left provides links for Diagnose and solve problems, Access Control (IAM), and Settings. The main area displays a table with one item:

	Name	Last modified	Access tier	Blob type	Size	Lease state
<input type="checkbox"/>	index.html	1/30/2026, 7:05:41 PM	Hot (Inferred)	Block blob	133 B	Available

The screenshot shows a web browser window with the address bar containing 'newsorage.z29.web.core.windows.net'. The page content displays the text 'Hello world' followed by a waving hand emoji.

Step 8: Configuring Lifecycle Management Policy

- Navigated to **Data management** → **Lifecycle management** in the storage account.
- Created a new lifecycle management rule named **MoveToCoolAfter100Days**.
- Applied the rule to **all blobs** in the storage account.
- Selected the **Blob type** as **Block blobs** and **Base blobs**.
- Configured the condition to trigger when blobs **have not been modified for 100 days**.
- Set the action to **move blobs to the Cool access tier**.
- Saved and updated the lifecycle management rule successfully.

A rule is made up of one or more conditions and actions that apply to the entire storage account. Optionally, specify that rules will apply to particular blobs by limiting with filters.

Rule name *
MoveToCoolAfter100Days

Rule scope *

Apply rule to all blobs in your storage account
 Limit blobs with filters

Blob type *

Block blobs
 Append blobs

Blob subtype *

Base blobs
 Snapshots
 Versions

Update **Previous** **Next**

The screenshot shows the Azure Storage Lifecycle Management interface for creating a new rule. The top navigation bar includes links for Home, newsorage, Lifecycle management, and Update a rule. The main area is titled 'Update a rule' and has tabs for 'Details' and 'Base blobs'. A note states: 'Lifecycle management uses your rules to automatically move blobs to cooler tiers or to delete them. If you create multiple rules, the associated actions must be implemented in tier order (from hot to cool storage, then archive, then deletion).'. Below this, a flowchart-like structure shows an 'If' condition 'Base blobs haven't been modified in 100 days' connected by a downward arrow to a 'Then' action 'Move to cool storage', also connected by a downward arrow to a '+ Add conditions' button.

TASK 3: Azure IoT Hub – Sensor Data Ingestion and Storage

Step 1: Preparing Blob Containers for IoT Data Storage

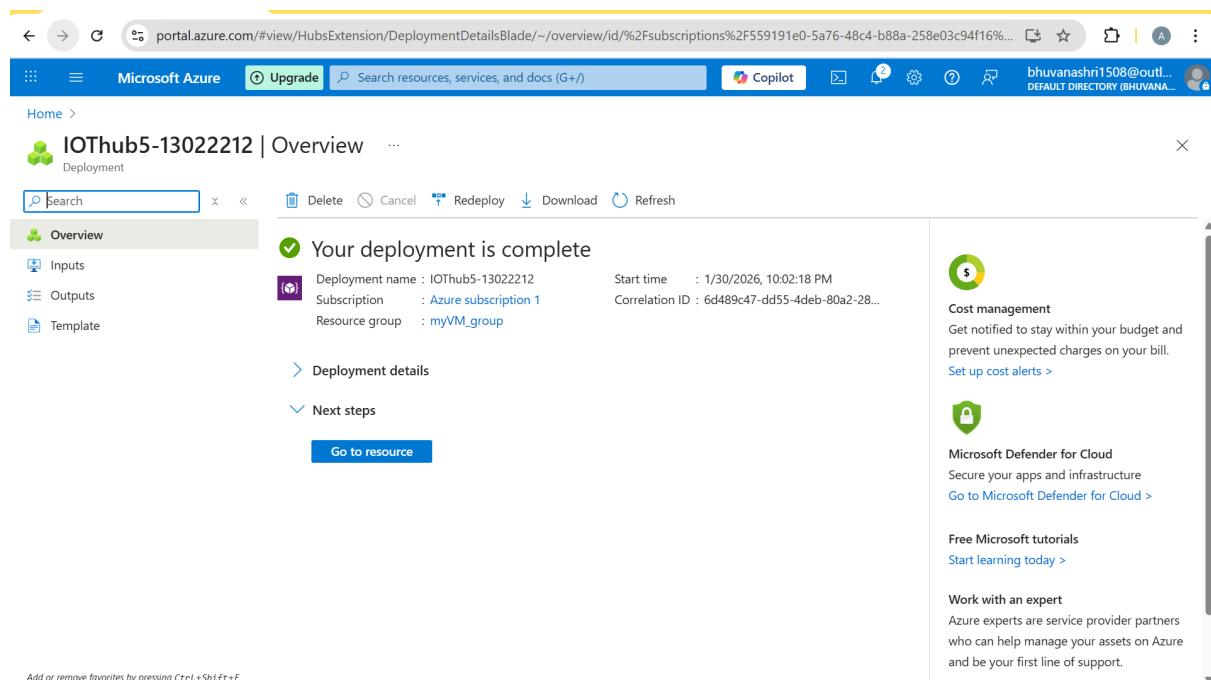
- Opened the **Storage Account → Containers** section in the Azure Portal.
- Verified the presence of multiple containers used for different purposes.
- Confirmed the **telemetry** container is created to store IoT sensor output data.
- Observed system-generated containers such as **\$logs** and **\$web** for logging and static website hosting.
- Ensured the **telemetry** container is in **Available** state and ready to receive routed IoT data.
- This step prepares the storage layer required for saving temperature and humidity data from the IoT Hub.

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes tabs for 'newsorage - Microsoft Azure', 'Red Apple Images – Browse 2.3', and 'Hello'. The main header bar has a 'Microsoft Azure' logo, an 'Upgrade' button, a search bar ('Search resources, services, and docs (G+/)'), and various account and settings icons. The left sidebar shows a tree view with 'Data storage' selected under 'Storage account'. The main content area displays a table of containers:

Name	Last modified	Anonymous access level	Lease state
\$logs	1/30/2026, 6:48:57 PM	Private	Available
\$web	1/30/2026, 6:48:57 PM	Private	Available
images	1/30/2026, 6:20:33 PM	Blob	Available
telemetry	1/30/2026, 9:47:18 PM	Private	Available

Step 2: Creating an Azure IoT Hub

- Initiated the creation of an **Azure IoT Hub** from the Azure Portal.
- Verified that the IoT Hub deployment completed successfully.
- Confirmed the IoT Hub name as **IOTHub5-13022212**.
- Associated the IoT Hub with the resource group **myVM_group**.
- Verified that the provisioning state shows **Your deployment is complete**.
- Accessed the IoT Hub resource to proceed with device configuration and message routing.



Step 3: Registering a Device in Azure IoT Hub

- Opened the **IoT Hub** → **Device management** → **Devices** section.
- Created and registered a new IoT device with the **Device ID: rpi-sim-01**.
- Verified the device type as **IoT Device**.
- Confirmed the device **status is Enabled**.
- Ensured the device authentication is configured successfully.
- This step registers the Raspberry Pi simulator as a device in the IoT Hub, enabling data transmission to Azure.

The screenshot shows the Microsoft Azure portal interface for managing IoT devices. The top navigation bar includes links for Home, IoT Hub, and IOHub5. The main title is "IOHub5 | Devices". On the left, there's a sidebar with options like Create, Manage view, Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, Resource visualizer, and Device management (with sub-options for Devices, IoT Edge, Configurations + Deployments, Updates, and Queries). A message at the top of the sidebar says, "You are viewing a new version of Browse experience. Click here to access the old experience." Below the sidebar, a search bar and a table list a single device: "rpi-sim-01". The table columns are Device ID, Type, Status, Last st..., Authe..., C2D ..., and Tags. The device details show it is an IoT Device, Enabled, and has 0 shared connections.

Step 4: Retrieving Device Connection Details

- Opened the registered IoT device **rpi-sim-01** in the IoT Hub.
- Viewed the **device authentication details**, including primary and secondary keys.
- Retrieved the **Primary Connection String** for the device.
- Confirmed that the device connection to the IoT Hub is **Enabled**.

- These credentials are used by the Raspberry Pi simulator to securely send

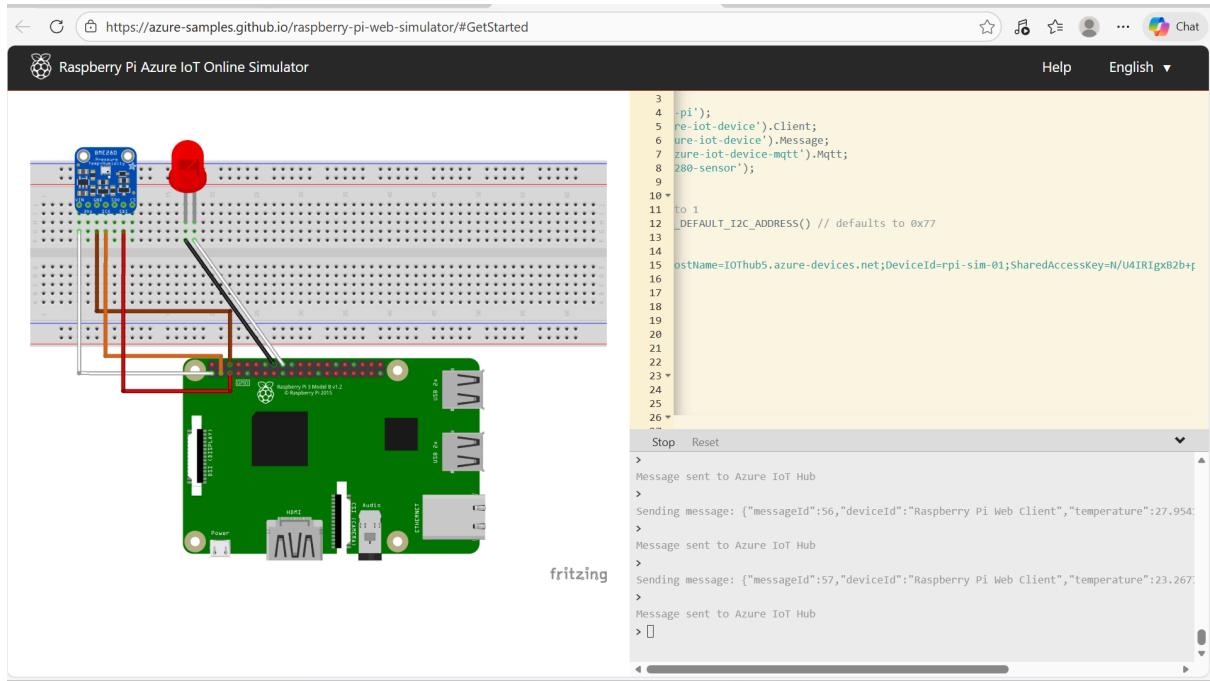
The screenshot shows the Azure IoT Hub Device blade for the device 'rpi-sim-01'. The device ID is set to 'rpi-sim-01'. The primary key, secondary key, primary connection string, and secondary connection string are all present with redacted values. Under 'Tags', it says 'No tags'. Under 'Enable connection to IoT Hub', the 'Enable' radio button is selected. Under 'Parent device', it says 'No parent device'. At the bottom, there are tabs for 'Module Identities' and 'Configurations', with a note: 'Please update to Standard SKU to utilize this feature.'

temperature and humidity data to Azure IoT Hub.

Step 5: Simulating Raspberry Pi Sensor Data

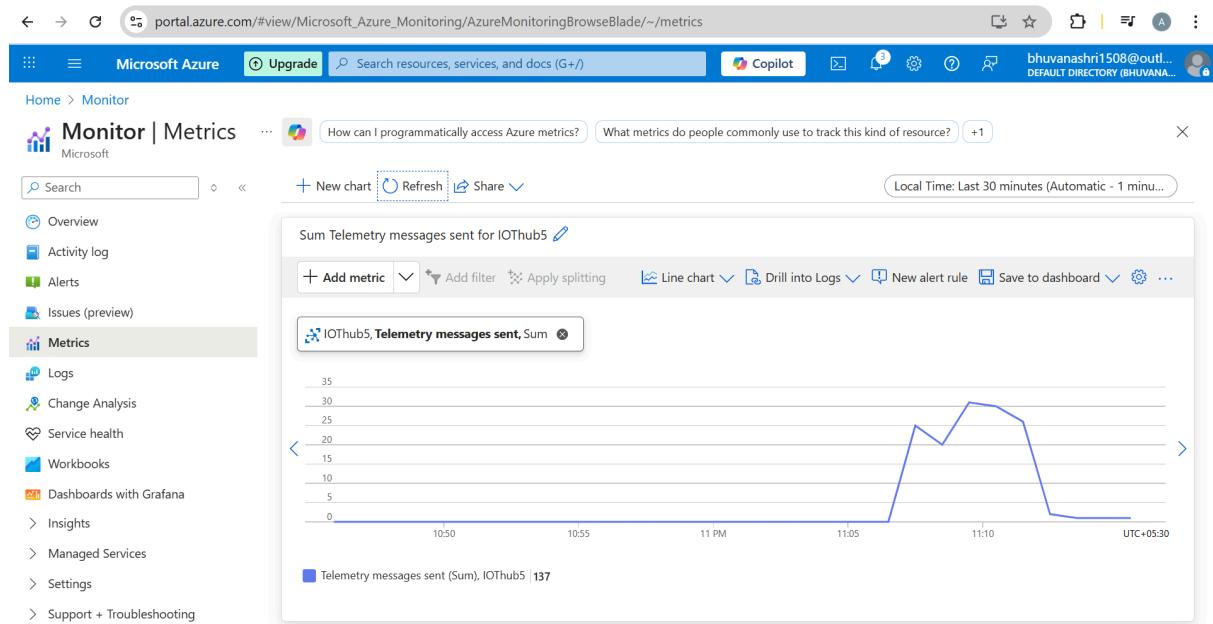
- Opened the **Raspberry Pi Azure IoT Online Simulator**.
- Configured the simulator using the **IoT Hub device connection string**.
- Simulated temperature and humidity sensor readings.
- Verified that telemetry messages are generated and sent to Azure IoT Hub.

- Confirmed successful message transmission in the simulator console.



Step 6: Monitoring IoT Telemetry in Azure

- Navigated to **Azure Monitor → Metrics** for the IoT Hub.
- Selected the **Telemetry messages sent** metric.
- Verified that telemetry data from the simulated device is being received.
- Observed message trends in the metrics graph, confirming active data flow.



Step 7: Configuring Message Routing from IoT Hub to Blob Storage

- Opened the **IoT Hub → Message routing** section.
- Created a custom route named **telemetry-to-blob**.
- Selected the **data source** as **DeviceMessages**.
- Configured the **routing query** as **true** to route all incoming messages.
- Selected the **custom endpoint** pointing to **Azure Blob Storage**.
- Verified that the route is **Enabled**.
- This step ensures that all telemetry data from the IoT device is automatically routed to Blob Storage.

The screenshot shows the Microsoft Azure portal interface for managing an IoT Hub. The URL in the address bar is <https://portal.azure.com/#@bhuvanashri1508outlook.onmicrosoft.com/resource/subscriptions/559191e0-5a76-48c4-b88a-258e03c94f16/resourceGroup...>. The page title is "IoT Hub5 | Message routing". The left sidebar shows the IoT Hub navigation path: Home > IoT Hub > IoTHub5. The main content area is titled "Message routing" under the "IoT Hub" section. A sidebar on the left has sections for "Access control (IAM)", "Hub settings", "Message routing" (which is selected), "Security settings", and "Shared access policies". A search bar at the top right contains the text "mess". A message box says "You have created one or more custom routes which would stop messages from flowing to the built-in endpoint. Click here to add a route to the built-in endpoint if you would like to keep the data flowing there." Below this is a table with the following data:

Name	Data Source	Routing Query	Endpoint	Enabled
telemetry-to-blob	DeviceMessages	true	telemetry-endpoint	true

Step 8: Verifying IoT Telemetry Data Stored in Blob Storage

- Opened the **telemetry** container in the Azure Storage Account.
- Navigated through the auto-generated folder hierarchy created by IoT Hub routing.
- Verified multiple **.avro** files generated from incoming telemetry data.
- Confirmed the blob type as **Block blob** with access tier **Hot (Inferred)**.
- Observed timestamps matching the telemetry transmission time.
- This confirms that temperature and humidity data from the simulated device is successfully stored in **Azure Blob Storage**.

The screenshot shows the Microsoft Azure Storage Container blade for a container named 'telemetry'. The left sidebar includes links for Overview, Diagnose and solve problems, Access Control (IAM), and Settings. The main area displays a list of blobs with the following details:

Name	Last modified	Access tier	Blob type	Size	Lease state
[...]					...
01.avro	1/30/2026, 11:33:15 PM	Hot (Inferred)	Block blob	1.09 KiB	Available
03.avro	1/30/2026, 11:35:15 PM	Hot (Inferred)	Block blob	1.09 KiB	Available
05.avro	1/30/2026, 11:37:15 PM	Hot (Inferred)	Block blob	1.09 KiB	Available
07.avro	1/30/2026, 11:39:18 PM	Hot (Inferred)	Block blob	1.49 KiB	Available

TASK 4: Azure SQL Database Creation and Power BI Integration

Step 1: Initiating Azure SQL Database Deployment

- Started the deployment of a new **Azure SQL Database** from the Azure Portal.
- Selected the **Azure subscription** and assigned the database to the resource group **myVM_group**.
- Triggered the creation of a **new SQL Server** along with the database.
- Verified that the deployment status shows “**Deployment is in progress**”, indicating successful initiation.
- This step sets up the backend database infrastructure required for further data analysis and reporting.

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes the Microsoft Azure logo, an Upgrade button, a search bar, and user account information. Below the header, the page title is "Microsoft.SQLDatabase.newDatabaseNewServer_27fcc9a393a04dfaa55bc | Overview". On the left, there's a sidebar with "Overview", "Inputs", "Outputs", and "Template" options. The main content area displays a message "Deployment is in progress" and a table of deployment details:

Resource	Type	Status	Operation
task4sqlserver...	Microsoft.Sql/servers	Accepted	Operation

Below the table, a note says "Add or remove favorites by pressing Ctrl + Shift + F". To the right, there are promotional cards for Microsoft Defender for Cloud, free tutorials, and expert support.

Step 2: Verifying Successful Azure SQL Database Deployment

- Monitored the deployment status of the Azure SQL Database in the Azure Portal.
- Confirmed that the deployment status shows "**Your deployment is complete**".
- Verified the database was created under **Azure subscription 1** and resource group **myVM_group**.
- Noted the successful completion timestamp and correlation ID for audit/reference.

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes 'Microsoft Azure', 'Upgrade', a search bar, and user information 'bhuvanashri1508@outlook.com'. The main content area displays a deployment overview for 'Microsoft.SQLDatabase.newDatabaseNewServer_27fcc9a393a04dfa55bc'. A green checkmark indicates 'Your deployment is complete'. Deployment details show: Deployment name: Microsoft.SQLDatabase.newDa..., Start time: 31/01/2026, 16:49:22; Subscription: Azure subscription 1; Correlation ID: bc32b213-bcbe-4710-96ef-3b1...; Resource group: myVM_group. Below this are sections for 'Deployment details' and 'Next steps', with a 'Go to resource' button. To the right, there are promotional cards for 'Cost management', 'Microsoft Defender for Cloud', 'Free Microsoft tutorials', and 'Work with an expert'.

Step 3: Login to Azure SQL Database using Query Editor

- Opened **Azure SQL Database Query Editor (Preview)** from the Azure Portal.
- Selected the database **Task4-AdventureWorksLT**.
- Logged in using **SQL Server / Microsoft Entra authentication**.
- Successfully connected to the Azure SQL Database.

The screenshot shows the 'Query editor (preview)' page for the 'Task4-AdventureWorksLT' database. The top navigation bar includes 'Microsoft Azure', 'Upgrade', a search bar, and user information 'bhuvanashri1508@outlook.com'. The main content area shows a welcome message: 'Welcome to SQL Database Query Editor'. It provides two authentication options: 'SQL server authentication' (Login: CloudSA0217c309, Password: [redacted]) and 'Microsoft Entra authentication' (Logged in as bhuvanashri1508@outlook.com, Continue as bhuvanashri1508@outlook.com). A large 'OK' button is at the bottom.

Step 4: Execute SQL Query and View Results

- Executed the SQL query **SELECT TOP 10 * FROM SalesLT.Customer**; using the Azure SQL Database Query Editor (Preview).
- The query was run successfully without any errors.
- Customer records from the SalesLT.Customer table were displayed in the results pane.

The screenshot shows the Microsoft Azure Query editor (preview) interface. The title bar reads "Task4-AdventureWorksLT (task4sqlserver-xyz/Task4-AdventureWorksLT) | Query editor (preview)". The main area displays a query window titled "Query 1" containing the following SQL code:

```
1 SELECT TOP 10 * FROM SalesLT.Customer;
```

Below the query window, the "Results" tab is selected, showing a table with the following data:

NameStyle	Title	FirstName	MiddleName	LastName
False	Mr.	Orlando	N.	Gee
False	Mr.	Keith		Harris

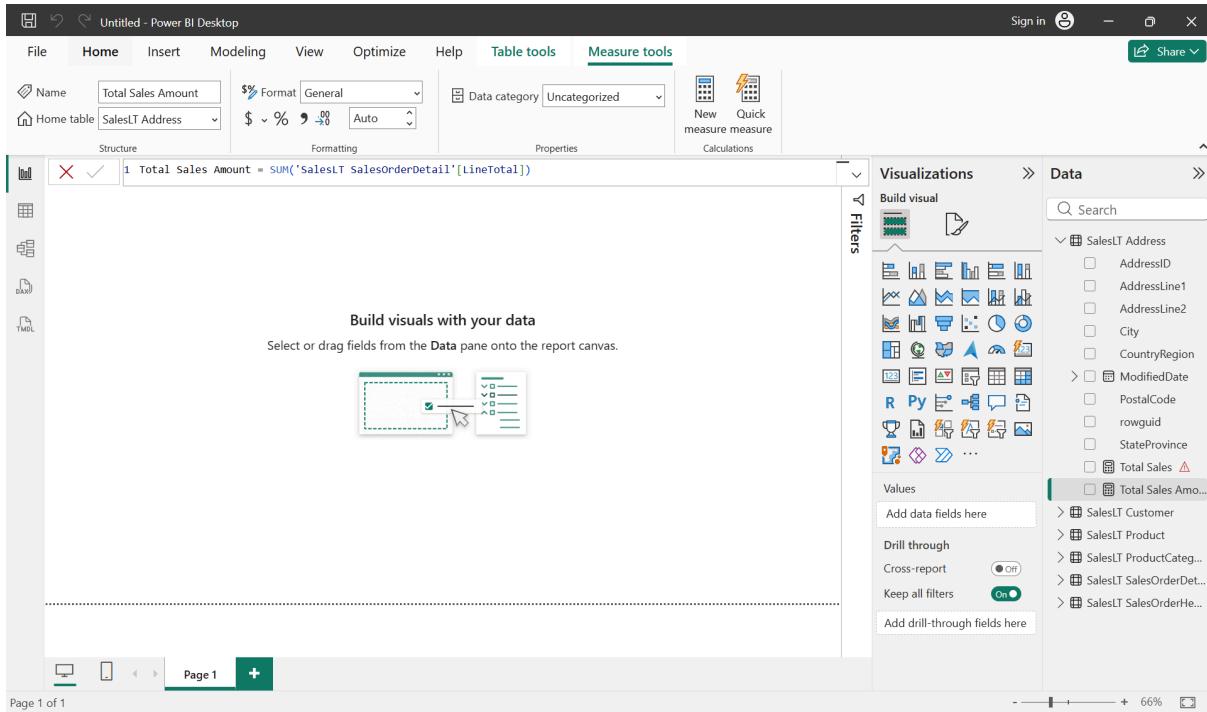
A status bar at the bottom indicates "Query succeeded | 0s".

task4sqlserver-xyz.database.windows.net -- server name

Step 5: Create a Measure for Total Sales Amount in Power BI

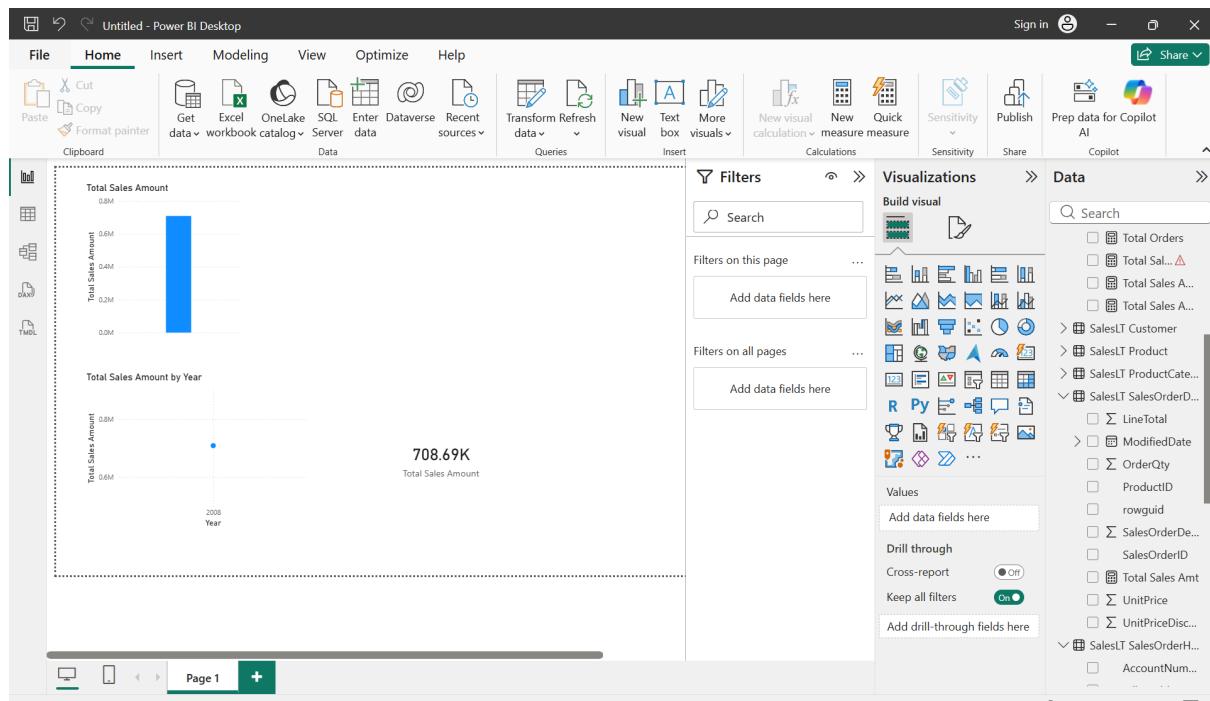
- Opened **Power BI Desktop** and loaded the AdventureWorksLT dataset.
- Created a new measure named **Total Sales Amount**.
- Defined the measure using the DAX expression
Total Sales Amount = SUM(SalesLT_SalesOrderDetail[LineTotal])
- The measure calculates the overall sales value from the sales order details table.

- This measure is used for building sales-related visualizations.



Step 6: Build Visualizations Using the Total Sales Measure

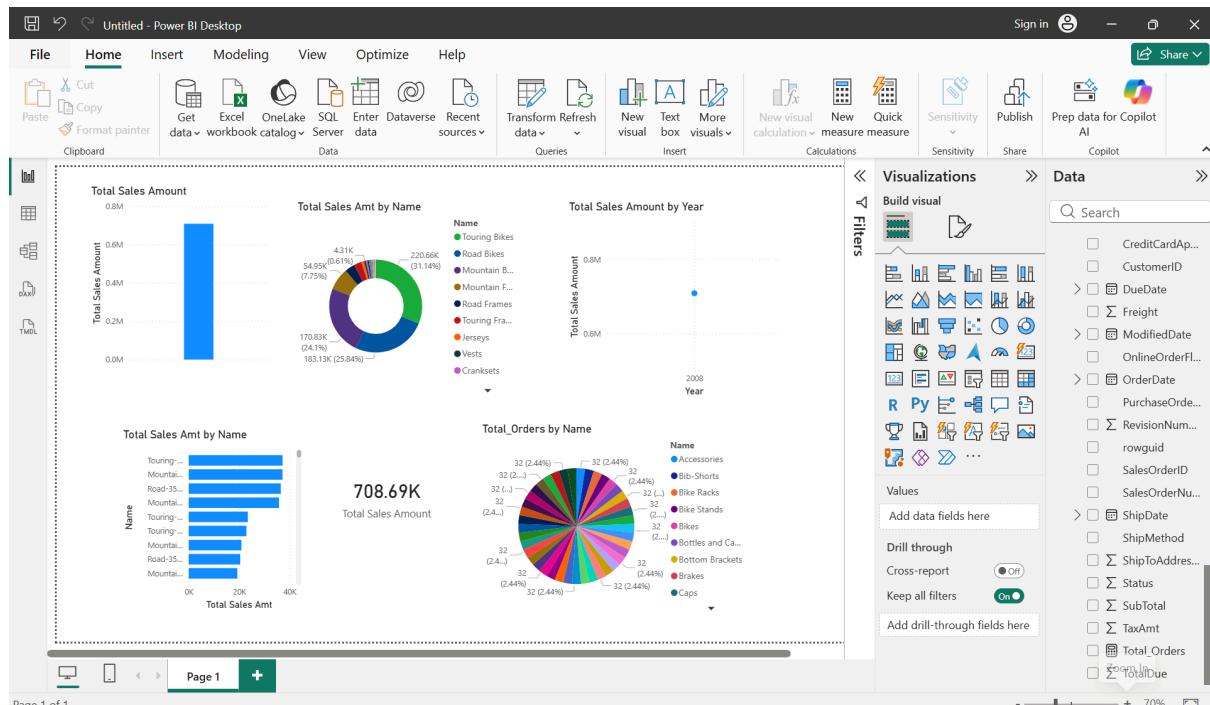
- Created visualizations using the **Total Sales Amount** measure.
- Displayed total sales using bar and card visuals.
- Visualized total sales trends (such as Total Sales Amount by Year).
- Verified that the visuals correctly reflect aggregated sales data.
- This confirms successful data modeling and visualization in Power BI.



Step 7: Create required visualizations in Power BI

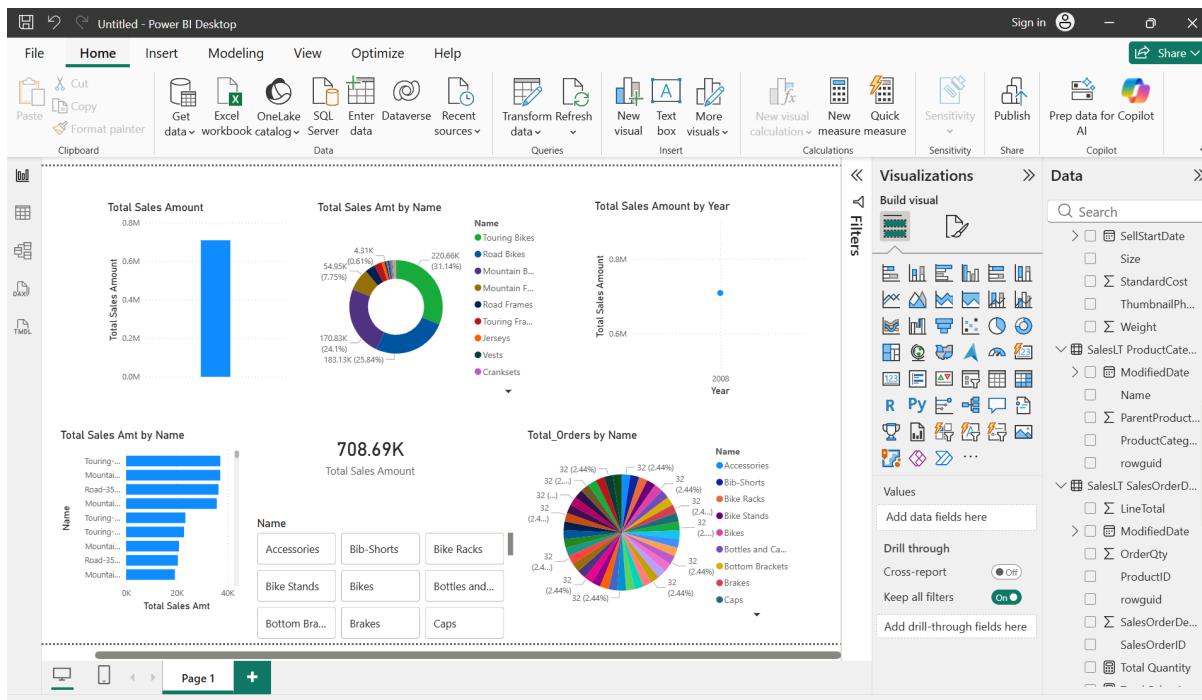
In Power BI Desktop, use the loaded AdventureWorksLT tables and created measures to build visualizations. Add the following visuals to the report canvas:

- A column chart showing **Total Sales Amount**
- A donut chart showing **Total Sales Amount by Product Name**
- A bar chart showing **Total Sales Amount by Product Name**
- A scatter or line chart showing **Total Sales Amount by Year**
- A pie chart showing **Total Orders by Product Name**
- A card visual displaying **Total Sales Amount**



Step 8: Format and finalize the Power BI dashboard

Adjust the layout and formatting of all visuals for clarity and presentation. Apply titles, legends, and labels to each chart. Align visuals neatly on the canvas and ensure consistent colors and sizing. Add slicers if required (for example, Product Name or Category) to enable interactive filtering. Save the Power BI report file (.pbix) after verifying that all visuals display correct data.



CONCLUSION

This project successfully demonstrates how cloud-based databases and business intelligence tools can be integrated to analyze and visualize business data effectively. Azure SQL Database enables secure and scalable data storage, while Power BI transforms the data into meaningful insights through interactive dashboards.

By applying SQL querying, data modeling, and visualization techniques, the project provides clear insights into sales trends and product performance. Overall, the project highlights the importance of cloud analytics in modern business environments and shows how Azure and Power BI can support informed decision-making.

