Lesson - 20 Homework

Question 1: How does Power BI handle large datasets in the Online Service, and what is the role of Premium Capacity in this?

Power BI Service uses in-memory compression to load datasets into RAM for fast query performance. Here's how it scales:

Standard Handling (Pro License):

1GB size limit per dataset

Runs on shared capacity (your reports compete with others for resources)

Uses DirectQuery for larger data (slower, queries the live database)

Premium Capacity's Role:

Removes the 1GB limit (supports up to 400GB datasets)

Provides dedicated hardware (your own guaranteed CPU/memory)

Enables Aggregations - stores summarized data in memory while keeping details in the source

Uses automatic load balancing to handle many concurrent users

Allows deeper compression and better memory management

The Bottom Line: Premium gives Power BI the dedicated resources and advanced features needed to handle large datasets efficiently, while standard Pro has strict limits for shared resource protection.

Question 2: What are the differences between Import mode, DirectQuery, and Live Connection in Power BI Service?

1. Import Mode (The Default & Most Powerful)
How it Works: You copy and load a snapshot of the data from the source into Power BI's highly compressed, in-memory engine (VertiPaq).

Pros:

Blazing Fast Performance: All interactions and calculations happen in-memory.

Full Feature Access: Unlocks all of Power BI's data modeling capabilities (DAX, complex relationships, calculated tables).

Works Offline: Once imported, you can work with the data without a connection to the source.

Cons:

Data Latency: Data is only as fresh as the last scheduled or manual refresh.

Size Limits: Bound by Power BI's dataset size limits (1GB on Pro, up to 400GB on Premium).

2. DirectQuery (For Real-Time & Large Data)
How it Works: No data is copied into Power BI. It maintains a live connection to the source database (e.g., SQL Server, Snowflake). Every time you filter, slice, or add a visual, Power BI generates and runs a native query directly on the source database.

Pros:

Real-Time Data: Always shows the current state of the source database.

Bypasses Size Limits: Can report on massive datasets that are too large to import.

Cons:

Performance Risk: Speed depends entirely on the source database's health and load. Can be slow with complex reports.

Limited Modeling: You cannot create calculated tables or perform complex Power Query transformations after the initial connection.

3. Live Connection (For Corporate Semantic Layers)
How it Works: A specialized form of DirectQuery that connects exclusively to an Analysis Services model (SQL Server Analysis Services, Azure Analysis Services) or a Power BI dataset.

Pros:

Leverages a Single Source of Truth: Uses a pre-built, corporate data model that is already optimized, secure, and governed.

Excellent Performance: Leverages the powerful Analysis Services engine.

Cons:

Zero Modeling in Power BI: You are a consumer of the model. You cannot add columns, create measures, or change relationships in your Power BI report. All modeling must be done in the external source.

Question 3: Explain deployment pipelines in Power BI Online. What stages do they include?

Deployment Pipelines in Power BI Online are a DevOps tool that lets you manage the lifecycle of your Power BI content (reports, datasets, dashboards) through development, testing, and production stages.

What Are Deployment Pipelines?
A structured framework in Power BI Service with dedicated stages for:

Development: Where you build and test new content

Test/UAT: Where business users validate the content

Production: Where end-users consume the final content

The 3 Stages
Development Stage

Purpose: "Sandbox" environment for building and initial testing

Users: Report developers, data modelers

Content: Work-in-progress reports, experimental features

Test Stage (UAT)

Purpose: Quality assurance and user acceptance testing

Users: Business stakeholders, QA testers

Content: Near-final versions for validation

Production Stage

Purpose: Live environment for end-users

Users: Report consumers, business decision-makers

Content: Final, approved, and stable content

Key Features & Benefits
One-Click Deployment: Move content between stages with a single click

Configuration Rules: Set different data sources for each stage (e.g., Dev database → Production database)

Comparison View: See differences between stages before deploying

Backward Deployment: Move content backward if needed

Access Control: Different permissions for each stage

Automation Support: Can be automated via PowerShell/REST APIs

How It Works
Assign Workspaces: Link each stage to a specific Power BI workspace

Develop Content: Build reports in your Development workspace

Deploy to Test: Click "Deploy" to move content to Test stage

Validate & Approve: Business users test in Test environment

Go Live: Deploy approved content to Production stage

Requirements
Premium License Needed: Requires Power BI Premium capacity (or Premium Per User)

Workspace Setup: Each stage must be assigned to a workspace

Permissions: Users need appropriate access rights for each stage

Bottom Line: Deployment Pipelines bring software development best practices to Power BI, ensuring controlled, reliable, and organized deployment of business intelligence content.

Question 4: How can Power BI Service integrate with Microsoft Teams or SharePoint for collaboration?

Power BI integrates deeply with Microsoft Teams and SharePoint to enable seamless collaboration and embed analytics directly into your daily workflow.

Integration with Microsoft Teams
1. Power BI App in Teams
Add the Power BI tab to any team channel

Pin entire reports or dashboards for team access

Team members can interact with reports without leaving Teams

2. Share Reports in Chat
Copy and paste Power BI links directly into Teams chats

Links unfurl into rich, interactive previews

Colleagues can view and discuss data side-by-side

3. Automated Alerts & Notifications
Subscribe to Power BI data alerts that post to Teams channels

Get notified when metrics hit thresholds

Power Automate flows can trigger Teams messages based on data changes

4. Collaborate with Teams Meetings
Share report windows during Teams meetings

Live present data insights to stakeholders

Annotate and discuss visuals in real-time

Integration with SharePoint
1. Embed Reports in SharePoint Pages
Use the "Power BI" web part in modern SharePoint pages

Publish to web option for public-facing dashboards

Secure embed for internal company portals

2. Two Embedding Methods:
SharePoint Online web part (recommended for internal sites)

iFrame embed (works with any SharePoint version)

3. Single Sign-On (SSO)
Users access embedded reports without re-authenticating

Permissions flow through from SharePoint

Row-Level Security (RLS) respects user identity

4. Centralized Content Management
Store PBIX files in SharePoint document libraries

Use version history for report development

Co-authoring capabilities for report teams

Collaboration Benefits
Contextual Discussions: Talk about data where work happens

Reduced Context Switching: No need to jump between apps

Governance & Control: Maintain security and permissions

Wider Reach: Expose reports to users who might not regularly visit Power BI Service

Setup Requirements
Power BI Pro license for users publishing and sharing content

Appropriate permissions in both Power BI and Teams/SharePoint

Modern SharePoint experience for web part integration

Power BI workspace access for Teams tabs

Bottom Line: These integrations make Power BI a collaborative analytics hub rather than a standalone tool, embedding data-driven decision making into your organization's daily workflow.

Question 5: What is the XMLA endpoint in Premium and how does it benefit developers or enterprise BI teams?

XMLA Endpoint in Power BI Premium is a gateway that allows external tools to communicate with Power BI datasets using the industry-standard XMLA protocol (same protocol used by SQL Server Analysis Services).

What It Is
A read-write connection point to Power BI datasets

Uses the standard Tabular Model protocol (XMLA)

Enables external tools to connect directly to datasets in the Power BI Service

Key Benefits for Developers & Enterprise BI Teams
1. Advanced Dataset Management
Use SQL Server Management Studio (SSMS) to:

View dataset metadata and partitions

Execute DAX queries for debugging

Monitor performance and metrics

2. Professional Development Tools
Use Tabular Editor for:

Metadata management (drag-and-drop measures, calculated columns)

Bulk operations (create/edit multiple measures at once)

C# scripting for advanced automation

Source control integration with Git

3. Enterprise Deployment Pipelines
Automate deployments using PowerShell scripts

CI/CD integration with Azure DevOps

Programmatic dataset refresh and management

4. External Application Integration
Custom applications can query Power BI datasets directly

Other Microsoft tools (Excel, Azure Analysis Services) can connect

Third-party BI tools can leverage Power BI datasets as a source

5. Advanced Monitoring & Optimization
Query performance analysis

Memory usage monitoring

Direct access to Dynamic Management Views (DMVs)

Practical Use Cases
Developer Workflow: Build a data model in Tabular Editor → Deploy via XMLA → Visualize in Power BI

Enterprise Scale: Manage 100+ measures across multiple datasets efficiently

DevOps Integration: Automated testing and deployment of data models

Performance Tuning: Identify and fix slow queries directly

Requirements
Power BI Premium capacity (or Premium Per User)

Read or Read/Write permissions enabled in the capacity admin settings

External tools like SSMS, Tabular Editor, DAX Studio

Bottom Line: XMLA endpoints transform Power BI from a self-service tool into an enterprise-grade analytics platform, enabling professional development practices, automation, and advanced management capabilities that enterprise BI teams require.

Question 6: Describe how usage metrics and audit logs work in Power BI Service.

Service, highlighting their distinct purposes.

Usage Metrics: The "What" - User Engagement & Report Performance
Usage Metrics answer the question: "How are my reports and dashboards being used?"

What They Track:
Report Opens: How many times a report was viewed.

Unique Viewers: How many individual users accessed it.

Sharing Activity: How often reports are shared or embedded.

Performance Data: Report load times and DAX query duration.

Popular Pages: Which tabs within a report are most viewed.

How to Access Them:
Go to your Workspace in the Power BI Service.

Select a report or dashboard.

Click on the "Usage Metrics" icon in the action bar.

Power BI automatically generates a pre-built report showing you the usage data for that specific content.

Key Benefits:
Identify Popular Content: See which reports provide the most value.

Find Unused Content: Discover and retire reports no one uses.

Performance Tuning: Identify slow reports that need optimization.

Justify ROI: Demonstrate the adoption and impact of your BI initiatives.

Audit Logs: The "Who, When, and Where" - Security & Governance
Audit Logs answer the question: "Who did what, when, and from where?" in your entire Power BI tenant.

What They Track:
User Sign-ins: Login attempts and failures.

Data Export Events: When users export data to PDF, Excel, or .CSV.

Permission Changes: When workspace access or report permissions are modified.

Admin Actions: Changes to tenant settings or capacities.

Content Lifecycle: Events like publishing a report, creating a workspace, or deleting a dashboard.

How to Access Them:
You must be a Power BI Admin or have the appropriate Office 365 admin role.

Go to the Microsoft 365 Admin Center (or the Purview Compliance Portal).

Navigate to Audit Log Search.

Filter by Activities, Users, Date Range, and specifically set the Service to Power BI.

Key Benefits:
Security Monitoring: Investigate suspicious activity or data breaches.

Compliance Reporting: Prove adherence to internal or external regulations.

Troubleshooting: Diagnose issues by seeing the sequence of user actions.

Governance: Monitor how Power BI is being used across the organization.

Question 7: How do you manage workspace access and permissions for different users?

The Two-Level Permission Model
Workspace permissions are managed at two main levels: Workspace Roles and App Permissions.

Level 1: Workspace Roles (For Developers & Contributors)
These control who can create, edit, and manage content inside the workspace.

Role Capabilities Typical User
Admin - Full control (add/users, delete workspace)
- Publish/update apps
- Update data sources & gateways BI Team Lead, Project Owner
Member - Add/edit/del content (reports, datasets)
- Can't manage user access Report Developer, Data Analyst
Contributor - Add/edit content they create
- Cannot publish apps or delete overall content Junior Analyst, Business User
Viewer - View only content in the workspace
- Can create personal bookmarks & comments Stakeholder, Reviewer
How to Assign:

Go to the workspace.

Click Access.

Add users/groups and assign a role.

Level 2: App Permissions (For End-Users)
These control who can view and use the published application.

Publish an App: When your content is ready, you publish an App from the workspace.

Grant Access: You can grant access to:

Entire organization

Specific individuals or Microsoft 365 Groups

Permissions:

View Permissions: Users can interact with the app but not edit it.

Build Permissions (Premium): Users can build their own reports on top of the datasets in your app.

How to Assign:

In your workspace, click "Create app" or update an existing one.

During publishing, go to the "Permissions" tab.

Add the end-users or security groups who need access.

Best Practices for Management
Use Microsoft 365 Groups (Recommended)

Create a group like "Sales-Analytics-Team".

Add the group as a Workspace Member.

Manage users in Azure AD/Teams; access syncs automatically.

Leverage Security Groups for Apps

For large audiences, assign app permissions to a security group (e.g., "All-Sales-Users") instead of individual users.

Apply the Principle of Least Privilege

Don't make everyone an Admin. Most users only need Viewer or Contributor roles.

Separate Development from Production

Use Deployment Pipelines with different workspaces (Dev, Test, Prod).

Grant Admin/Member in Dev, Viewer in Test, and no direct access to Prod (users only get the app).

Use Viewer for Stakeholder Feedback

Instead of giving them the app, add them as a Workspace Viewer to see content early and leave comments.

Question 8: How can data governance be enforced in Power BI Service?

1. Tenant-Level Governance (The Foundation)
Managed by Power BI Administrators in the Admin Portal.

Export Controls: Restrict users from exporting data to Excel, CSV, or PowerPoint.

Sharing Limits: Control who can publish to the web or share content externally.

Workspace Creation: Restrict who can create new workspaces to prevent sprawl.

Certification & Promotion: Enable a "Promotion" and "Certification" process for datasets and reports, allowing administrators to endorse trusted, official data sources.

2. Data Security & Access Control
Row-Level Security (RLS): The cornerstone of data security. Implement dynamic RLS so users only see the data they are permitted to see (e.g., a salesperson only sees their region's data).

Workspace Roles: Use the principle of least privilege:

Admins/Members for developers.

Viewers for stakeholders who only need to see content.

App-Based Distribution: Instead of giving users direct workspace access, publish curated Apps to control exactly what content is consumed.

3. Data Lineage & Impact Analysis
Lineage View: Use Power BI's built-in lineage view to see how datasets, reports, and dashboards are connected. This is crucial for understanding dependencies before making changes.

Data Sensitivity Labels: Apply Microsoft Purview Information Protection labels from the admin center. These labels (e.g., "Public," "Confidential," "Highly Confidential") can:

Encrypt data at rest.

Apply watermarks to reports.

Enforce access policies based on user and device.

4. Deployment & Lifecycle Management
Deployment Pipelines: Enforce a strict Development → Test → Production workflow. This ensures proper testing and approval before content reaches end-users.

APIs & Automation: Use the Power BI REST API with Azure DevOps for full CI/CD (Continuous Integration/Continuous Deployment) automation, ensuring standardized and audited deployments.

5. Monitoring & Auditing
Audit Logs: Continuously track user activity in the Microsoft 365 Compliance Center. Monitor for events like data exports, permission changes, and report accesses.

Usage Metrics: Regularly review usage metrics to identify unused content (which can be archived) and monitor the performance and adoption of key reports.

Question 9: What are the limitations of Row-Level Security when using DirectQuery or Live Connection?

1. Performance Impact & Query Overhead

How it works: In DirectQuery/Live, the RLS filter is translated into a WHERE clause in the native query sent to the source database.

The Limitation: Every single visual interaction (slicing, filtering, etc.) triggers a new query to the source system that includes the RLS logic. This can place a significant additional load on your source database, especially with complex security rules or many concurrent users.

2. Complexity of DAX Rules
How it works: Your RLS rules are written in DAX, but the source database understands SQL.

The Limitation: Power BI must translate your DAX-based RLS rules into the source system's SQL dialect. Complex DAX functions may not translate efficiently or at all, leading to query failures or performance degradation. You must stick to simple, filter-based DAX for reliable cross-platform execution.

3. Limited Data Source Support for Dynamic RLS
Dynamic RLS relies on the USERPRINCIPALNAME() or CUSTOMDATA() DAX functions to filter data based on the logged-in user.

The Limitation: Not all data sources can handle the context switch these functions require. While major relational databases (SQL Server, Oracle) and Azure services typically support it, other or legacy sources might not, causing the security to fail.

4. Double-Hop Security in Live Connection
How it works: In a Live Connection to Analysis Services (SSAS), the RLS is defined and enforced in the source model, not in Power BI.

The Limitation (Double-Hop): The user's identity must be passed from Power BI Service, through the gateway, to the source SSAS model. This requires Kerberos Constrained Delegation to be correctly configured, which is a complex and often problematic setup for IT teams.

5. No "Test as Role" Feature in Service
In Import mode, you can easily test RLS roles directly in the Power BI Service.

The Limitation: For DirectQuery and Live Connection, the "Test as Role" feature in the Service is disabled. The only way to test security is to log in as different users, making validation much more cumbersome.

6. Model Logic Limitations
Live Connection Specific: Since you cannot modify the data model in a Live Connection, all RLS rules must be defined and maintained in the source Analysis Services model. Power BI developers lose the ability to manage security within the Power BI context.

Question 10: Explain how you can refresh a dataset via Power Automate or REST API.

Method 1: Using Power Automate (No-Code/Low-Code)
This is ideal for triggering a dataset refresh based on an event or schedule without writing code.

Common Triggers:

Schedule: Refresh every day at 6 AM.

File Arrival: Refresh when a new file lands in SharePoint/OneDrive.

Email Received: Refresh when an email with a specific subject arrives.

Button Click: Manual trigger from the Power Automate mobile app.

Step-by-Step Flow:

Create a new Flow in Power Automate.

Choose your Trigger (e.g., "Recurrence" for a schedule).

Add the "Refresh a dataset" Action:

Search for "Power BI" and select the "Refresh a dataset" action.

Configure the Action:

Workspace: Select the workspace containing your dataset.

Dataset: Select the specific dataset you want to refresh.

Save and Test.

Example: Refresh after a Data Pipeline Completes

Trigger: "When an HTTP request is received" (your data pipeline calls this URL when it finishes).

Action: "Refresh a dataset" in Power BI.

Result: Your Power BI dataset refreshes automatically the moment your upstream ETL process completes.

Method 2: Using the REST API (Pro-Developer Approach)
This gives you full programmatic control and is used for custom applications, DevOps scripts, or advanced automation.

The Core API Call:

http
POST https://api.powerbi.com/v1.0/myorg/groups/{workspaceId}/datasets/{datasetId}/refreshes
You need to get the workspaceId (Group ID) and datasetId.

Step-by-Step Process:

Authentication:

You must authenticate using an Azure AD App Registration.

This service principal must have the required permissions (e.g., Member or Admin role) in the target Power BI workspace.

Get Access Token:

Use a client secret or certificate to get an OAuth2 token from Microsoft Identity Platform.

Make the API Call:

Send a POST request to the refresh API endpoint with the access token in the header.

Example using PowerShell:

powershell
```
# 1. Get Parameters
$tenantId = "your-tenant-id"
$clientId = "your-app-client-id"
$clientSecret = "your-client-secret"
$workspaceId = "your-workspace-guid"
$datasetId = "your-dataset-guid"

# 2. Get Access Token
$body = @{
    grant_type    = "client_credentials"
    client_id     = $clientId
    client_secret = $clientSecret
    resource      = "https://analysis.windows.net/powerbi/api"
    scope         = "https://analysis.windows.net/powerbi/api/.default"
}
$response = Invoke-RestMethod -Uri "https://login.microsoftonline.com/$tenantId/oauth2/token" -Method POST -Body $body
$accessToken = $response.access_token

# 3. Trigger Dataset Refresh
$headers = @{
    'Authorization' = "Bearer $accessToken"
    'Content-Type' = 'application/json'
}
$refreshUri = "https://api.powerbi.com/v1.0/myorg/groups/$workspaceId/datasets/$datasetId/refreshes"
Invoke-RestMethod -Uri $refreshUri -Method POST -Headers $headers
```
Advanced API Features
Refresh History: GET .../datasets/{datasetId}/refreshes

Enhanced Refresh (Premium): Use a JSON payload to specify parameters like:

notifyOption: Send an email on refresh failure.

type: Perform a "Full" or "Incremental" refresh.

Refresh via XMLA Endpoint (Premium): For ultimate control, use the Tabular Model Scripting Language (TMSL) via the XMLA endpoint to script and execute precise refresh commands.

Comparison

Feature	Power Automate	REST API
Ease of Use	Easy (Visual Designer)	Advanced (Requires Coding)
Control	Basic (Trigger Refresh)	Full Control (Type, Notifications, History)
Integration	Best with Microsoft Ecosystem	Works with any system (Python, .NET, etc.)
Cost	Requires Power Automate Premium license if using premium connectors	Free (just the cost of your script/app)
Authentication	User-delegated or Service Principal	Primarily Service Principal (App Registration)

Bottom Line: Use Power Automate for simple, event-driven refreshes within the Microsoft cloud. Use the REST API for complex, custom, or large-scale automation integrated into your IT infrastructure.