USN | | | | | | | | | | |

Internal Assessment Test1–September2025

Sub:	DATAENGINEERING AND MLOPS					Sub Code:	BAD714C	Branch:	AIML/CSE AIML	
Date:	**29/09/25**	Duration:	90min	MaxMarks:	50	Sem/Sec:	VII/A,B&C		OBE	
Answer any FIVE FULL Questions								**MARKS**	**CO**	**RBT**
1	Explain the stages of the Data Engineering Life Cycle, What are the main challenges faced at each stage? Explain various data maturity stages in data engineering							10	CO1	L2
2	Explain the trade-offs between TypeA and TypeB data engineering approaches. How do they align with different organizational needs and data maturity Stages?							10	CO1	L2
3	Apply the concept of "Good Data Architecture" to design a simple architecture for an online library system that serves 10,000 users.							10	CO2	L3
4a	Explain the concept of FinOps in cloud cost optimization.							5	CO2	L2
4b	Explainthedifferencebetweendataarchitectureandenterprisearchitecture With examples.							5	CO2	L2
5a	DifferentiateGITandDVC,Withoneexample,applythecommandsofDVCtooland Explain the requirement of GIT, during the data version control							5	CO3	L3
5b	Define feature selection and dimensionality reduction .How do they help Mitigate the curse of dimensionality?							5	CO3	L1
6	What arethechallenges aMLOpsengineerfaceduringdeploymentat production level.Writedownhowtomitigatetheriskinvolvedin Mlops.							10	CO3	L1

Faculty Signature CCI Signature HOD Signature

USN

Internal Assessment Test 1 – November 2025

Sub:	**Data Engineering and MLOPS**					Sub Code:	BAD714C	Branch:	AIML/CSE AIML
Date:	**29/09/2025**	Duration:	90 min's	Max Marks:	50	Sem/Sec:	VII /A,B CSEAIML(A)		OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1.	**Explain the stages of the Data Engineering Life Cycle, What are the main challenges faced at each stage? Explain various data maturity stages in data engineering.** **Data Engineering Life Cycle stages [4]** **Data Collection** – Gathering raw data from various sources like databases, APIs, sensors, or files. **Data Ingestion** – Importing data into a storage system (like a data lake or warehouse) using batch or real-time pipelines. **Data Storage** – Organizing and storing data in databases, warehouses, or lakes for easy access and scalability. **Data Processing** – Cleaning, transforming, and structuring data to make it usable for analysis or machine learning. **Data Integration** – Combining data from multiple sources to create a unified, consistent dataset. **Data Analysis & Modeling** – Using analytical tools or ML models to extract insights and patterns. **Data Visualization & Reporting** – Presenting data insights through dashboards, charts, or reports for decision-making. **Monitoring & Maintenance** – Tracking data quality, pipeline performance, and making updates as systems evolve. *Main challenges faced at each stage [4]* **Data Collection :** Inconsistent or missing data from sources ,Data in different formats , Access restrictions or API limits **Data Ingestion :** Handling large data volumes in real time,Data duplication or loss during transfer ,Network and latency issues. **Data Storage** : Choosing the right storage system (SQL, NoSQL, data lake, etc. ,Managing storage costs ,Ensuring data security and backup . **Data Processing** : Data quality issues (errors, outliers, duplicates) , Complex transformation logic ,Processing speed and scalability **Data Integration:** Schema mismatches between sources ,Maintaining consistency across systems , Handling updates and changes in source data. **Data Analysis & Modeling:** Dealing with biased or incomplete data ,Selecting appropriate analytical models ,Ensuring reproducibility and validation Data Visualization & Reporting: Choosing the right visualization tools ,Misinterpretation of results ,Keeping dashboards up to date with new data **Monitoring & Maintenance:** Detecting and fixing pipeline failures ,Managing concept drift (model degradation) ,Ensuring data governance and compliance Data Maturity Stages : Ad-hoc , Repeatable , Defined , Managed , Optimized [2]	10	CO1	L2
2	**Explain the trade-offs between Type A and Type B data engineering approaches. How do they align with different organizational needs and data maturity Stages?** **Type A data engineering approaches**: Type A Data Engineering focuses on **data for analysis, reporting, and business intelligence (BI)**. The goal is to make data **clean, consistent, and easily accessible** for decision-making — not necessarily real-time applications.	10	CO1	L2

	Type B data engineering approaches: Type B Data Engineering focuses on **building data systems that power real-time applications, AI/ML models, and data-driven products**. The goal is to make data **fast, scalable, and actionable** for intelligent systems — not just reports. **Trade offs between Type A and Type B data engineering: Both** differ mainly in purpose and complexity. **Type A** focuses on analytics, reporting, and business intelligence using structured, historical data processed in batches. It is simpler, cost-effective, and best suited for organizations in early to mid data maturity stages. In contrast, **Type B** supports real-time applications, AI, and machine learning by handling diverse data types and using real-time or event-driven architectures. While it offers high scalability and enables automation, it is more complex, expensive, and requires advanced technical skills. In summary, Type A enables data-driven decisions, whereas Type B powers data-driven actions — and mature organizations often combine both for balance. **Different organizational needs and data maturity Stages** Organizations in the **early stages of data maturity** usually adopt **Type A**, focusing on building reliable data pipelines for reporting, dashboards, and business insights. Their priority is to ensure data quality, consistency, and accessibility rather than real-time analytics. As organizations **mature**, their data needs evolve toward **real-time decision-making**, automation, and AI-driven products — areas where **Type B** excels. Type B suits **high-maturity organizations** that have established data governance, advanced infrastructure, and skilled teams capable of handling large-scale, real-time, and unstructured data. In essence, **Type A supports foundational analytics and decision-making**, while **Type B enables intelligent, automated, and real-time data applications** in advanced data-driven organizations.			
3.	**Apply the concept of "Good Data Architecture" to design a simple architecture for an online library system that serves 10,000 users.** **Good Data Architecture Principles** 1. **Scalability:** Should handle growth in users, books, and activity. 2. **Modularity:** Separate components for catalog, users, borrowing, and analytics. 3. **Data Consistency & Integrity:** Ensure accurate book availability, user data, and borrowing history. 4. **Security & Privacy:** Protect user data and sensitive information. 5. **Performance:** Fast search, book checkout, and recommendation queries. 6. **Maintainability:** Easy to update and extend with new features. **Proposed Simple Architecture for Online Library** 1. **User Layer (Frontend):** Web/mobile apps where users search books, borrow, return, and rate. 2. **Application Layer (Backend)** Handles business logic: authentication, borrowing rules, search, recommendations. 3.**Data Layer:** **Relational Database (SQL):** Store structured data like users, books, borrow/return logs.**NoSQL Database (Optional):** Store semi-structured data like book reviews or logs.**Search Engine (Elasticsearch):** Fast book searches and recommendations. 3. **ETL / Analytics Layer:** Batch or stream processing pipelines to analyze borrowing patterns, popular books, and system performance. 4. **Security & Governance:** Role-based access control for admins and users. Encryption for sensitive user data.	10	CO2	L3

Data Flow Example

1. User searches for a book → Frontend → Backend (Catalog Service) → Search Engine → Results displayed.
2. User borrows a book → Backend (Borrowing Service) → Relational DB updated → Analytics pipeline logs activity.
3. Recommendations generated via analytics pipeline and returned to users.

4.a Explain the concept of FinOps in cloud cost optimization. 5 CO2 L2

FinOps is a **collaborative approach** that brings together **finance, operations, and engineering teams** to manage and optimize cloud spending.

How FinOps Optimizes Cloud Costs

- **Right-Sizing Resources:** Adjust compute/storage instances to actual usage.
- **Reserved Instances & Savings Plans:** Pre-purchase capacity to reduce on-demand costs.
- **Auto-Scaling:** Automatically scale resources up/down based on demand.
- **Tagging & Cost Allocation:** Attribute costs to departments/projects for transparency.
- **Eliminating Waste:** Delete unused resources, idle VMs, or obsolete storage

4.b Explain the difference between data architecture and enterprise architecture With examples. 5 CO2 L2

DATA ARCHITECTURE	ENTERPRISE ARCHITECTURE
Data management: storage, pipelines, quality	Organization-wide IT systems and processes
Limited to data-related components	Broad: business, applications, technology, and data
Ensure accurate, accessible, and secure data	Align IT systems and processes with business objectives
Data warehouse for sales analytics	Full IT landscape: e-commerce platform, inventory, cloud infrastructure, and data warehouse

5.(a)	Differentiate GIT and DVC , With one example , apply the commands of DVC tool and explain the requirement of GIT , during the data version control GIT ✓ Version control for code and small text files. ✓ Not suitable for large files (>100 MB) ✓ Local or remote Git repository ✓ Tracks changes in code (commits) DVC ✓ Version control for large data files, datasets, and ML models ✓ Efficiently tracks large files without storing them in Git ✓ Links data to remote storage (S3, GCP, Azure, etc.) ✓ Tracks changes in data and model files, along with pipelines ✓ Teams can share datasets and model versions alongside code **Example** Suppose you have a machine learning project: • Code files: train.py, model.py → managed with Git • Dataset: data/large_dataset.csv → managed with DVC DVC Commands and Example Workflow 1. Initialize DVC in a project dvc init 2. Add a dataset to DVC dvc add data/large_dataset.csv • Creates a .dvc file (large_dataset.csv.dvc) that tracks the dataset. 3. Commit the DVC tracking file with Git git add data/large_dataset.csv.dvc .gitignore git commit -m "Track dataset with DVC" • Git tracks the .dvc file (metadata), not the large dataset itself. 4. Push dataset to remote storage dvc remote add -d myremote s3://my-bucket/data dvc push • The dataset is stored remotely; DVC keeps track of versions. 5. Pull dataset from remote (for collaborators) dvc pull		CO3	L3
5.(b)	**Define feature selection and dimensionality reduction. How do they help**	5	CO3	L2

mitigate the curse of dimensionality?

Feature Selection

Definition:

Feature selection is the process of **choosing a subset of the most relevant features** (variables) from the original dataset while **ignoring irrelevant or redundant features**.

Purpose:

- Improve model performance
- Reduce over fitting
- Lower computational cost

Example:

In a dataset predicting house prices, you may select only size, location, and age of the house as features, ignoring less important ones like color of the mailbox.

2. Dimensionality Reduction

Definition:

Dimensionality reduction is the process of **transforming high-dimensional data into a lower-dimensional space**, while **retaining most of the important information**.

Techniques:

- **PCA (Principal Component Analysis):** Projects data onto a smaller set of uncorrelated components.
- **t-SNE, UMAP:** Non-linear methods for visualization of high-dimensional data.

Example:

Reducing a dataset with 100 features to 10 principal components that explain 95% of the variance.

3. Mitigating the Curse of Dimensionality

The **curse of dimensionality** refers to problems that arise when the number of features is very large:

- Data becomes sparse, making it harder for models to find patterns.
- Increased risk of overfitting.
- Higher computational cost.

How Feature Selection and Dimensionality Reduction Help:

1. **Reduce the number of features:** Makes data less sparse and easier to analyze.
2. **Remove irrelevant/redundant information:** Improves model accuracy and generalization.
3. **Lower computational complexity:** Faster training and inference

6.	What are the challenges a MLOps engineer face during deployment at Production level. Write down how to mitigate the risk involved in Mlops.	10	CO3	L1

Challenges in Production-Level ML Deployment

1. **Data Drift & Concept Drift:**
   Problem: Incoming data or patterns change over time, causing model performance to degrade.
2. **Scalability & Performance:**
   Problem: Model may fail to handle high traffic or large volumes of data efficiently.
3. **Model Versioning & Rollback:**
   Problem: Difficult to manage multiple versions of models and revert if a new model fails.
4. **Reproducibility:**
   Problem: Inability to reproduce training experiments due to inconsistent environments or data.
5. **Monitoring & Logging:**
   Problem: Lack of continuous monitoring can lead to unnoticed failures or accuracy drops.
6. **Infrastructure & Deployment Complexity:**
   Problem: Integrating ML models with existing systems, handling dependencies, and containerization.
7. **Security & Compliance:**
   Problem: Sensitive data may be exposed; models may violate regulations
8. **Cost Management:**
   Problem: Inefficient resource usage can increase cloud or hardware costs.

Mitigating Risks in MLOps Deployment

1. **Continuous Monitoring:**
   Track model performance, accuracy, latency, and data quality in real-time.
2. **Automated Testing & Validation:**
   Test models on new data before full deployment. Include unit tests, integration tests, and regression tests.
3. **Version Control:**
   Use Git/DVC to track code, data, and model versions; maintain clear rollback mechanisms.
4. **Scalable Infrastructure:**
   Use containerization (Docker, Kubernetes) and auto-scaling to handle variable load.
5. **Data & Model Governance:**
   Implement access controls, encryption, and compliance checks for sensitive data.
6. **CI/CD Pipelines for ML:**
   Automate the training, testing, and deployment processes to reduce human error.
7. **Resource Optimization:**
   Monitor cloud costs, optimize inference pipelines, and right-size resources.
8. **Feedback Loops:**
   Continuously collect new data to retrain models and adapt to changing patterns.