Z5008: Big Data Practical Capstone Specification & Reference Key

Big Data Lab (MTech Core) · Labs 1–15 · innocent@iitmz.ac.in

Practical Exam Format: Duration: 2 Hours 40 Minutes · Dockerized Local Stack · 100 Marks. Design and verify an end-to-end streaming-to-MLOps Medallion pipeline.

Capstone Scenario: Zanzibar Mobile Money Fraud Lakehouse

Build a reproducible data and ML pipeline ingesting real-time payment events from Kafka topic `payments_raw` (`txn_id, user_id, merchant_id, region, amount_tzs, event_time`) into Bronze, Silver, and Gold Delta/Iceberg tables, training a fraud classifier tracked in MLflow, and exposing Prometheus health metrics.

Required Engineering Tasks & Rubric Breakdown [100 Marks]

Component	Weight	Acceptance Criteria
1. Lakehouse Medallion	25 pts	Partitioned Bronze/Silver Parquet+Delta, idempotent `MERGE`, `ZORDER`
2. Structured Streaming	25 pts	Event-time watermark (2 min), 5-min tumbling window, checkpointed sink
3. Airflow & Governance	20 pts	Acyclic DAG with retries, XCom URI passing, Iceberg field-ID rename
4. MLlib / MLflow / Obs.	30 pts	Pipeline estimator/transformer, MLflow stage gate, Prometheus p99 < 50 ms

Task 1. Silver Deduplication & Tuning: Write Spark logic deduplicating late retries on `txn_id` via Delta `MERGE`, handling 70% skew on `region='Unguja'`, and scheduling `OPTIMIZE/VACUUM`.

Task 2. Stateful Streaming: Explain watermark progression and checkpoint recovery if a Spark worker crashes mid-batch.

Task 3. Airflow & Iceberg: Show how an Airflow batch reconciliation job safely adds `risk_tier` in Iceberg without rewriting Bronze Parquet files.

Task 4. MLflow & Prometheus: Specify the exact promotion gate guarding `Staging` → `Production` and Prometheus PromQL alert rules.

Reference Architecture & Worked Solution Key

Task 1. Silver Idempotent MERGE & Skew Mitigation: Execute `DeltaTable.forPath(spark, silver_path).alias("t")` chaining `.merge(dedup_batch.alias("s"), "t.txn_id = s.txn_id")` with `.whenNotMatchedInsertAll().execute()`. Enable Spark 3 AQE (`spark.sql.adaptive.enabled=true`, `skewJoin.enabled=true`) or salt `region` with `0..15` buckets during regional aggregations before final reduction. Schedule daily `OPTIMIZE silver ZORDER BY (user_id)` to coalesce small micro-batch files into 256 MB Parquet files, and weekly `VACUUM silver RETAIN 168 HOURS`.

Task 2. Structured Streaming Watermark & Fault Recovery: `stream.withWatermark("event_time", "2 minutes")` paired with `.groupBy(F.window("event_time", "5 minutes"), "region")` and `.agg(F.count("*").alias("txns"), F.sum("amount_tzs").alias("vol"))`. At watermark $W = \max(\text{event_time}) - 2$ min, windows ending strictly before W finalize in `Append` mode and evict RocksDB state safely. On worker crash, Spark restarts from the durable WAL offset and state snapshot stored under `checkpointLocation`, guaranteeing end-to-end exactly-once sink state with Delta's transactional commit log.

Task 3. Airflow DAG & Iceberg Field-ID Evolution: Chain `verify_kafka_lag » silver_compact » gold_features » validate_dq`. Keep XCom payloads strictly to small S3/Delta partition URIs (`{{ ds }}`), enabling `retries=2` with idempotent partition overwrite/merge. In Apache Iceberg, running `ALTER TABLE gold ADD COLUMN risk_tier STRING` updates only the metadata JSON schema tree with a fresh integer `field_id`; older Parquet data files stay untouched and read back `NULL` safely for `risk_tier`.

Task 4. MLflow Gate & Prometheus SLA Alerting: Log run parameters, Delta commit version hash, and held-out PR-AUC / F1 inside `mlflow.start_run()`. A CI gate verifies `f1 >= 0.85` and `p99_latency_ms < 50` before calling `transition_model_version_stage(..., stage="Production", archive_existing_versions=True)`. Expose BentoML counters (`requests_total{status}`) and histograms (`inference_ms_bucket`), alerting in Grafana when `histogram_quantile(0.99, rate(inference_ms_bucket[5m])) > 50` or Kafka consumer lag rises across 3 consecutive scrapes.

Submission Checklist: Submit cleanly structured Python/PySpark modules, Docker Compose file, passing `unittest` output (`test_z5008_all.py`), and exported MLflow run summary metrics.