Practical Lab Assignment — Smart City with Artificial Intelligence (Select any one of the tasks listed below and submit it. Submitting more than one task is encouraged)

1. Aim

Apply AI techniques to solve real-world smart-city problems using geospatial, sensor or urban data. Build at least one working prototype (model + visualization) and document results.

2. Learning Objectives

- Acquire real urban datasets (IoT / open data / satellite / camera feeds).
- Preprocess spatio-temporal data for ML/DL.
- Train and evaluate ML/DL models for at least one application (classification, regression, segmentation or forecasting).
- Deploy results as visualizations: static maps, dashboards (Streamlit/Flask), or GIS layers (GeoTIFF/GeoJSON).
- Understand ethical, privacy and deployment considerations for smart city AI.

3. Tools / Software (suggested)

- Python (pandas, numpy, scikit-learn, matplotlib, geopandas)
- Deep learning: TensorFlow/Keras or PyTorch
- Geospatial: rasterio, shapely, fiona, folium, QGIS
- Earth Engine or Sentinel/Landsat (optional)
- Web: Streamlit / Flask for dashboard
- Optional: Docker, Google Colab / Drive for execution

4. Dataset Suggestions

(use any publicly available dataset)

- City open data portals (NYC, London, Singapore, Barcelona) — traffic, 311, parking.
- Traffic datasets: METR-LA, PEMS, Traffic4Cast (Kaggle/competitions).
- Air quality: OpenAQ, local sensor networks.
- Satellite imagery: Sentinel-2 (GEE), Landsat.
- CCTV / surveillance: synthetic or anonymized video frames (ensure privacy).
- Mobility: taxi trip records (NYC TLC), bike-share data.
- Waste management: smart bin datasets or synthetic.

5. Task Tracks (choose one primary + optional extras)

Track A — Smart Traffic

Beginner: Short-term traffic volume forecasting using historic counts (regression).
Intermediate: Congestion classification (low/medium/high) from sensor + weather data.
Advanced: Real-time adaptive signal control simulation (use RL or heuristic), predict optimal timings and show improvement metrics.

Deliverables (example):

- Data ingestion & cleaning notebook.
- Forecasting / classification model (code + trained model).
- Visual dashboard showing time series forecasts and heatmap of congestion.
- Short report: methodology, evaluation (MAE/RMSE/accuracy), limitations.

Track B — Smart Energy & Buildings

Beginner: Predict building energy consumption (regression).
Advanced: Demand forecasting + demand response scheduling (optimize HVAC setpoints to reduce peaks).
Deliverables: Model, simulation of savings, visualization of predicted vs actual load, short write-up.

Track C — Air Quality / Environmental Monitoring

Beginner: Spatial interpolation of sensor readings (ML).
Advanced: Spatio-temporal forecasting of AQI (LSTM, ConvLSTM) and hotspot mapping.
Deliverables: Prediction maps (GeoTIFF / folium heatmap), model metrics, analysis of major features.

Track D — Smart Waste & Logistics

Beginner: Predict bin fill levels (classification/regression) using historical fill events.
Advanced: Route optimization for collection vehicles using VRP solvers + real-time constraints.
Deliverables: Collection route planner (demo), comparison of route cost/time with baseline.

Track E — Public Safety & Computer Vision

Beginner: Object detection in street camera images (YOLO / Faster R-CNN) for counting vehicles/pedestrians.
Advanced: Anomaly detection in video (crowd density spikes, unusual movement) + alert demo.
Deliverables: Detection/alert demo, evaluation (precision/recall), privacy & ethics section.

6. Step-by-Step Instructions (example flow)

1. **Define scope:** Choose track & dataset..
2. **Data acquisition:** Download data (or use Earth Engine for imagery). Document source, date ranges, licenses.
3. **Exploratory Data Analysis (EDA):** Visualize distributions, missing values, spatial coverage.
4. **Preprocessing:**
    - Time series: resampling, imputation, feature engineering (lags, rolling stats).
    - Spatial data: reprojection, clipping, resampling, NDVI (if imagery).
    - Images: resizing, normalization, augmentation for training.
5. **Modeling:** Select appropriate ML/DL model. Train, validate, tune hyperparameters. Use cross-validation where applicable.

6. **Evaluation:** Use relevant metrics (MAE/RMSE for regression, F1/precision/recall for classification, IoU for segmentation). Visualize confusion matrices, residuals, maps.
7. **Deployment/Visualization:** Create a simple dashboard (Streamlit/folium) or generate GeoTIFF/GeoJSON deliverables.
8. **Ethics & Limitations:** Discuss privacy, bias, data quality, edge cases.
9. **Report & Presentation:** Submit code, notebook, README, model weights,

7. Required Deliverables

1. **Code:** Jupyter notebooks / Python scripts (well-commented).
2. **Trained model checkpoints** (or instructions to reproduce training).
3. **Visualizations:** maps, graphs, or a running dashboard link (if hosted).
4. **Report (PDF, max 6 pages):** intro, data, methods, results, discussion, conclusion.
5. **README** with step-by-step run instructions and dependencies.