Big Data Processing and Optimization

Learning Outcomes

After completing this unit, students will be able to:

- Explain the concept of Big Data and its characteristics (volume, velocity, variety).
- Identify key sources of environmental Big Data (satellites, sensors, open repositories).
- Understand major challenges in storing, cleaning, and processing large-scale datasets.
- Apply appropriate tools for Big Data processing (Google BigQuery, Spark, Hadoop, etc.).
- Optimize data queries and workflows for performance and cost efficiency.
- Visualize large datasets using dashboards and interpret results for environmental insights.
- Design and execute a scalable workflow for processing, analyzing, and visualizing environmental datasets.

Introduction

Environmental science has entered an era in which data volumes, complexity, and speed of generation far exceed what traditional data management and analysis methods were designed to handle. The term **Big Data** refers to datasets that are so large, fast-changing, or diverse that they require advanced computational tools, distributed systems, and new analytical approaches to extract meaningful information. In environmental science, Big Data is not simply about size; it is about continuous data streams, multiple data formats, and strong spatial and temporal dimensions that must be analyzed together.

Big Data is commonly described through several key characteristics, often referred to as the “V” properties:

- Volume – massive amounts of data, ranging from terabytes to petabytes, generated daily by satellites, sensor networks, and simulations.
- Velocity – high-speed data generation, such as real-time weather stations, air quality monitors, and ocean buoys.
- Variety – diverse data types, including numerical measurements, images, videos, time series, text-based reports, and geospatial layers.

- Veracity – varying data quality, uncertainty, and reliability, which are especially important in environmental measurements.
- Value – the potential to extract actionable knowledge for climate modeling, ecosystem protection, disaster prevention, and policy-making.

In environmental science, Big Data plays a crucial role in understanding complex Earth systems, detecting long-term trends, and supporting evidence-based decisions for sustainability and environmental protection.

Environmental Data Sources

Environmental Big Data originates from a wide range of sources, each contributing unique perspectives on natural and human-influenced processes.

Satellite data is one of the largest contributors. Earth observation satellites continuously collect information on land use, sea surface temperature, atmospheric composition, vegetation health, ice cover, and cloud dynamics. These datasets are typically high-resolution, multi-spectral, and global in coverage, making them extremely valuable but also computationally demanding.

In-situ sensors (in situ measurements require that the instrumentation is located directly at the point of interest and in contact with the subject of interest) provide ground-truth measurements and high-frequency data. These include weather stations measuring temperature, humidity, pressure, and precipitation; oceanographic buoys monitoring salinity, currents, and chlorophyll levels; air quality sensors detecting pollutants such as PM2.5, NO_2, and ozone; river and soil sensors tracking water levels, moisture, and nutrient content etc. Such sensors often operate continuously, generating streaming data that must be processed in near real time.

Global open databases further enrich environmental research. Organizations and initiatives such as meteorological agencies, space agencies, and international research consortia publish open-access datasets related to climate, biodiversity, oceanography, and atmospheric science. These datasets allow researchers to combine local measurements with global-scale observations, enabling comparative and long-term studies. Together, these sources form a complex, interconnected data ecosystem that defines Big Data in environmental science.

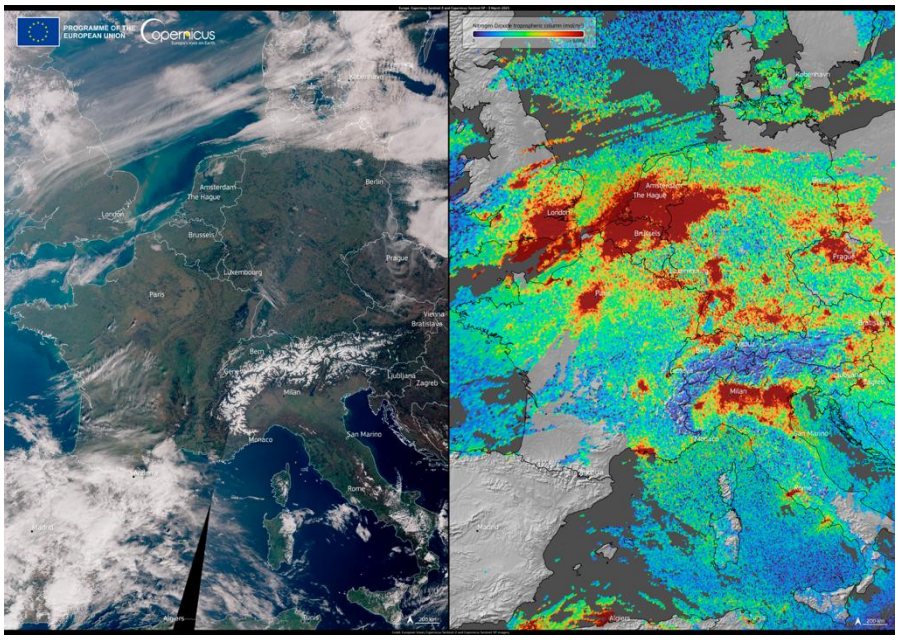

Figure 6.1 Copernicus Sentinel-5P satellite on 3 March 2025, highlighting nitrogen dioxide (NO_2) concentrations.

The Big Data Lifecycle

Managing environmental Big Data involves a structured process known as the Big Data lifecycle (Figure 6.2), which spans several interconnected stages.

Data collection: data is acquired from satellites, sensors, field measurements, simulations, and databases. This stage must consider sampling frequency, spatial resolution, and data formats to ensure compatibility and scientific relevance.

Data storage: due to the size and complexity of environmental datasets, storage often relies on distributed systems, cloud platforms, and specialized databases optimized for time-series and geospatial data. Efficient storage is essential for scalability and long-term accessibility.

Data cleaning and preprocessing: environmental data frequently contains missing values, noise, outliers, or inconsistent measurements. Cleaning involves error detection, gap filling, unit standardization, and temporal or spatial alignment. This step is critical, as poor data quality can lead to misleading conclusions.

Data processing: this stage transforms raw data into structured and analyzable forms. This may include aggregation, filtering, resampling, and integration of multiple datasets. In Big Data contexts, processing is often parallelized to handle large volumes efficiently.

Data analysis: analytical methods range from statistical analysis and trend detection to machine learning and predictive modeling. In environmental science, analysis is used to identify patterns, assess risks, and simulate future scenarios.

Data visualization: this stage translates complex datasets into intuitive maps, charts, dashboards, and animations. Effective visualization is essential for communicating results to scientists, decision-makers, and the public, especially when dealing with spatial and temporal phenomena.

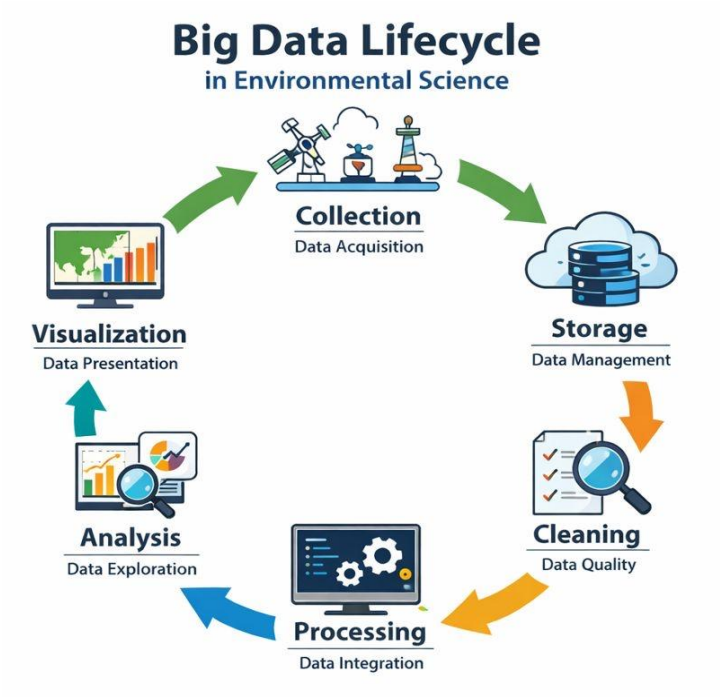

Figure 6.2 Big Data lifecycle

Key challenges in Environmental Big Data

Despite its potential, environmental Big Data presents several significant challenges. Heterogeneity is a major issue, as datasets differ in format, resolution, scale, and uncertainty. Combining satellite imagery with sensor readings and historical records requires careful harmonization.

Speed (velocity) poses challenges when data arrives continuously and must be processed in real time, such as during extreme weather events or pollution alerts. Also, data volume can overwhelm traditional computing systems. Long-term climate records and high-resolution satellite imagery demand scalable infrastructure and efficient algorithms.

Data quality and uncertainty are particularly critical in environmental applications. Sensor failures, calibration errors, and missing data are common, and uncertainty must be explicitly managed to ensure reliable scientific conclusions.

Big Data Formats and Storage Types

One of the simplest and most widely used big data formats is CSV (Comma-Separated Values). CSV files store data in plain text, where each row represents a record and columns are separated by commas. Due to their simplicity, CSV files are easy to generate, inspect, and exchange across platforms. They are commonly used for tabular data such as weather observations, pollution measurements, and statistical summaries. However, CSV files lack built-in metadata support, do not handle hierarchical data well, and become inefficient for very large datasets.

JSON (JavaScript Object Notation) is a semi-structured data format designed to represent hierarchical and nested data. JSON is particularly well suited for data exchanged through web APIs, IoT devices, and cloud services. In environmental monitoring systems, sensor networks frequently transmit data in JSON format, where measurements, timestamps, and device identifiers are grouped together in structured objects. While JSON is human-readable and flexible, it is less efficient than binary formats when handling massive datasets due to its verbosity.

To address performance and scalability limitations, modern Big Data systems often rely on columnar binary formats such as Parquet. Apache Parquet is optimized for analytical workloads and is commonly used in distributed processing systems like Apache Spark and cloud data warehouses. By storing data column-wise rather than row-wise, Parquet enables faster query execution, better compression, and reduced storage costs. This format is particularly effective for large historical datasets such as climate records or long-term environmental measurements.

In scientific domains, especially climate science and oceanography, NetCDF (Network Common Data Form) is a widely adopted format. NetCDF is specifically designed to store multidimensional data such as temperature, salinity, or atmospheric pressure across space and time. Unlike generic formats, NetCDF includes rich metadata that describes variables, units, and coordinate systems, making it highly suitable for reproducible scientific research.

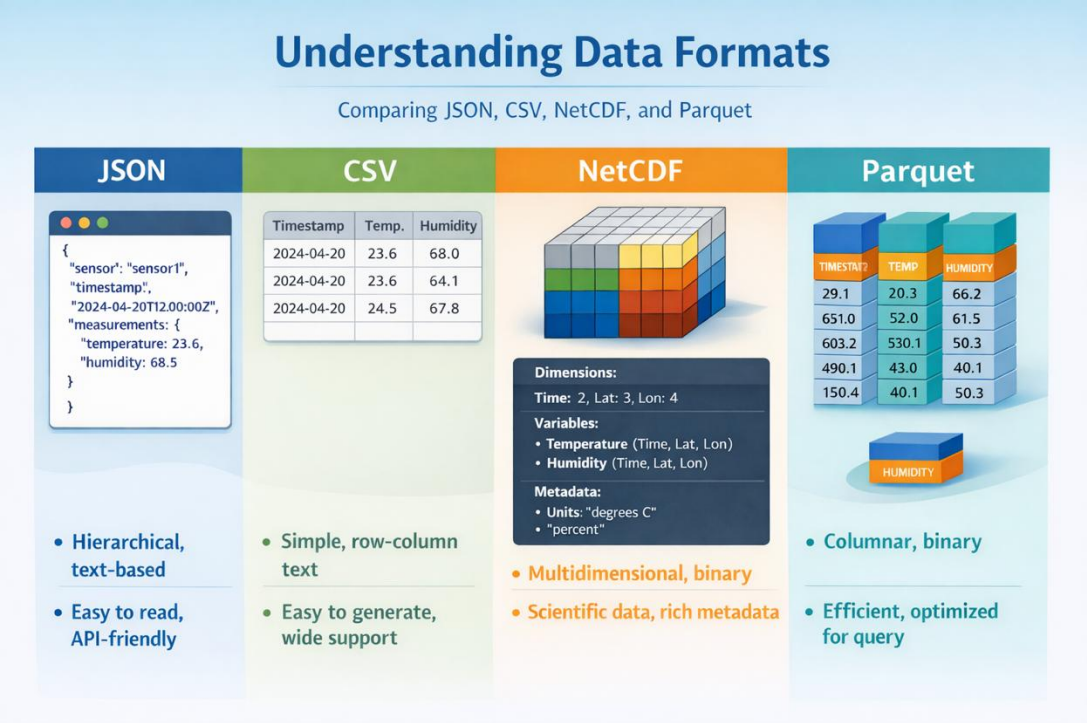

Figure 6.3 Understanding data formats – JSON, CSV, NetCDF and Parquet

Beyond data formats (Figure 6.3), Big Data systems depend on different storage types. Traditional local storage, such as files stored on personal computers or laboratory servers, is simple to manage but limited in scalability and reliability. As datasets grow in size and complexity, local storage becomes insufficient for collaborative and large-scale analysis.

Big Data processing requires coordinated software ecosystems capable of managing storage, computation, and resource allocation across many machines. One of the earliest and most influential ecosystems is Apache Hadoop, which introduced the concept of distributed storage (HDFS) combined with distributed computation through the MapReduce programming model. Hadoop enabled organizations to process datasets that were previously too large for single machines.

While Hadoop laid the foundation for Big Data, newer frameworks such as Apache Spark have gained widespread adoption due to their improved performance and flexibility. Spark supports in-memory computation, which significantly accelerates data processing tasks. It also provides high-level APIs for data analysis, machine learning, and graph processing. In environmental analytics, Spark is often used to process large sensor datasets or satellite imagery archives.

In recent years, cloud platforms have transformed the Big Data landscape by offering managed services that eliminate the need for manual infrastructure management. Google Cloud Platform (GCP) – Figure 6.4, Amazon Web Services (AWS), and Microsoft Azure provide integrated ecosystems that combine storage, processing, analytics, and visualization tools. These platforms allow researchers to focus on data analysis rather than system administration.

Cloud-based Big Data ecosystems are particularly attractive for interdisciplinary domains such as environmental science, where data volumes are high and computational demands vary over time. They support elastic scaling, meaning resources can be increased or decreased dynamically based on workload requirements, optimizing both performance and cost.

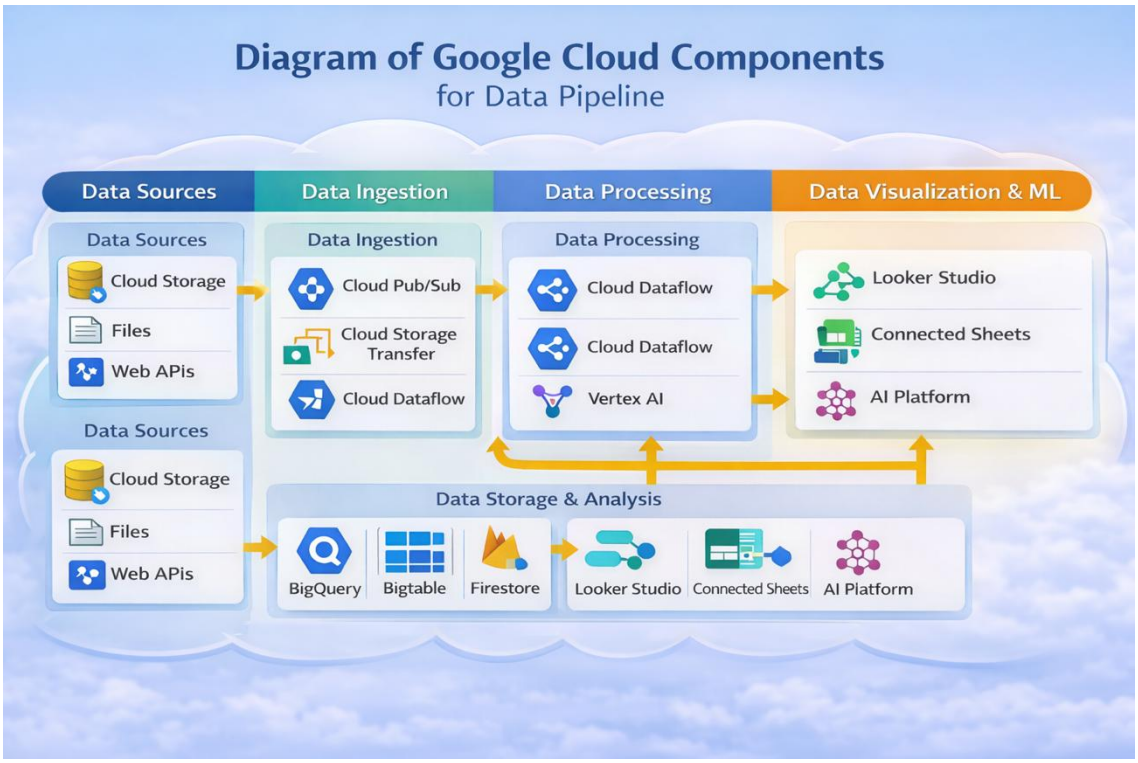

Figure 6.4 Google Cloud components

Data Preprocessing and Cleaning at Scale

Raw data is rarely ready for immediate analysis. In practice, Big Data often contains missing values, inconsistencies, duplicates, and measurement errors. Data preprocessing and cleaning are therefore critical steps in any Big Data workflow, especially when dealing with environmental or IoT data streams.

OpenRefine is a widely used open-source tool for interactive data cleaning and transformation. It allows users to explore datasets, detect anomalies, standardize values, and correct errors without requiring programming expertise. OpenRefine is particularly useful for preparing datasets obtained from public repositories or field studies.

Trifacta Wrangler, now known as Google Cloud Dataprep, extends these capabilities to cloud-scale datasets. It provides a visual, rule-based interface that enables users to define data transformation steps that can be executed at scale. This is especially useful when preparing large datasets stored in cloud environments such as Google Cloud Storage or BigQuery.

For users with programming experience, Pandas, a Python library, offers powerful data manipulation capabilities. Pandas is commonly used in combination with Jupyter notebooks and Google Colab to clean, filter, and transform datasets before large-scale processing. While Pandas is best suited for medium-sized datasets, it plays a crucial role in exploratory data analysis and prototyping.

Distributed and Parallel Processing Concepts

At the core of Big Data analytics lies the concept of distributed and parallel processing. Instead of processing data sequentially on a single machine, Big Data systems divide datasets into smaller chunks that can be processed simultaneously across multiple nodes.

MapReduce (Figure 6.5) is a way to process very large datasets by splitting the work into many small tasks that can run at the same time. The main reason it became popular is that it lets you handle “too-big-for-one-computer” data without needing one extremely powerful machine. Instead, you use many ordinary machines (or cloud computers) and divide the dataset among them. This is especially useful when the data is stored across multiple files or servers, such as logs, sensor readings, satellite tiles, or long-term environmental archives.

A helpful way to understand MapReduce is to focus on the roles of the two functions. The Map function is responsible for transforming raw input into intermediate pieces of information that are easy to combine later. It does not try to “finish” the problem; it prepares structured building blocks (key–value pairs). The Reduce function then combines all intermediate pieces that share the same key into a final answer. Many important Big Data tasks can be expressed in this form: counting, summing, grouping, finding averages, detecting maxima/minima, or building per-location statistics.

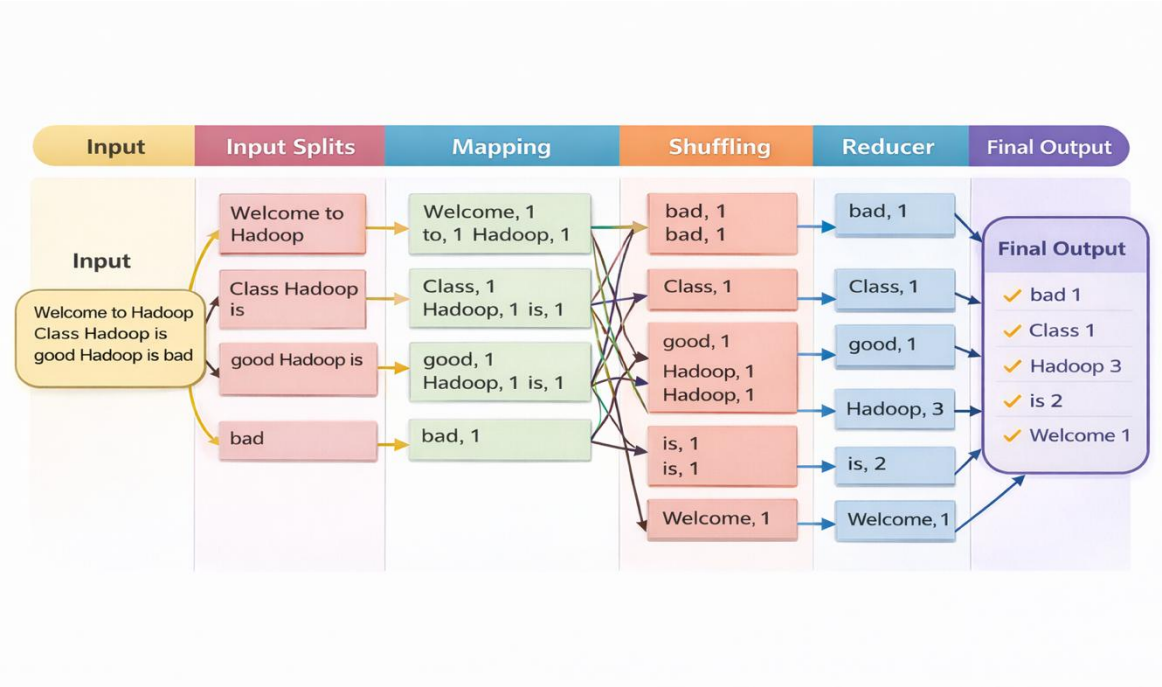

Figure 6.5 MapReduce operation

The “Shuffle” step (sometimes treated as part of the system rather than a user-defined step) is critical because it guarantees that all items with the same key end up together. In practice, shuffle is also one of the most expensive phases because it often requires moving data across the network between machines. For example, if you map a huge dataset into keys like “station_id” or “city,” the system must send all records for each station to the same reducer. Because network transfer is slower than memory operations, good MapReduce design tries to keep intermediate data small and keys well-balanced so no single reducer becomes overloaded.

One major advantage of MapReduce is fault tolerance. In large distributed systems, it is normal for some machines to fail. MapReduce frameworks (such as Hadoop) are designed so that if one worker crashes, its portion of the work can be automatically reassigned to another worker, and processing continues. This matters in real environmental applications where workflows may run for hours on huge data volumes; reliability becomes as important as speed.

In environmental and air-quality analytics, MapReduce-style thinking appears in many common tasks even if users don’t explicitly write MapReduce code. For example, imagine you have years of PM2.5 measurements from many monitoring stations. A Map step might emit (station_id, pm25_value) for each measurement. Then the Reduce step could compute per-station averages, daily maxima, or counts of days exceeding a threshold.

Today, many modern systems (like Spark, BigQuery under the hood, and Dataflow pipelines) don't require you to manually implement MapReduce, but the concept remains a powerful model. It explains why Big Data tools emphasize operations like group by, aggregate, and distributed joins—they are variations of the same core idea: map data into meaningful categories, shuffle it so related items meet, and reduce it into summaries that humans can interpret.

Indexing, Query Optimization, and Power Management

As datasets grow larger, efficient data access becomes increasingly important. Indexing is a technique that improves query performance by allowing systems to locate relevant data quickly without scanning entire datasets. In cloud data warehouses such as Google BigQuery, indexing and partitioning strategies significantly reduce query execution time and cost.

Query optimization involves selecting efficient execution plans for analytical queries. Modern Big Data systems automatically optimize queries by minimizing data movement, exploiting parallelism, and leveraging caching mechanisms. These optimizations are particularly important in cloud environments, where computational resources are billed based on usage.

In addition to performance considerations, power and energy management have become critical concerns. Large-scale data centers consume significant amounts of energy, prompting the development of energy-efficient hardware and scheduling algorithms. Cloud providers increasingly invest in renewable energy and energy-aware resource allocation, making cloud-based Big Data solutions more sustainable than traditional on-premise infrastructures.

Use Case: Cloud-Based Air Quality Data Processing and Analysis Using Google Tools

For the present use case, the OpenAQ dataset available via Google Cloud Marketplace/ BigQuery is selected as the primary data source. The dataset provides global air-quality observations (e.g., PM2.5, PM10, NO_2, O_3) compiled from multiple data providers and thousands of monitoring locations. Individual records typically include spatiotemporal metadata (country, city, monitoring site and often geographic coordinates), the measured pollutant/ parameter, the corresponding value and unit, and a timestamp. These characteristics make the dataset well suited to large-scale environmental analytics, including time-series trend analysis, spatial hotspot identification, and dashboard-based reporting using cloud-native tools such as BigQuery and Looker Studio. More broadly, OpenAQ aims to ingest, aggregate, harmonize, and disseminate interoperable real-time and historical air-quality data, thereby improving usability and comparability across heterogeneous sources and formats.

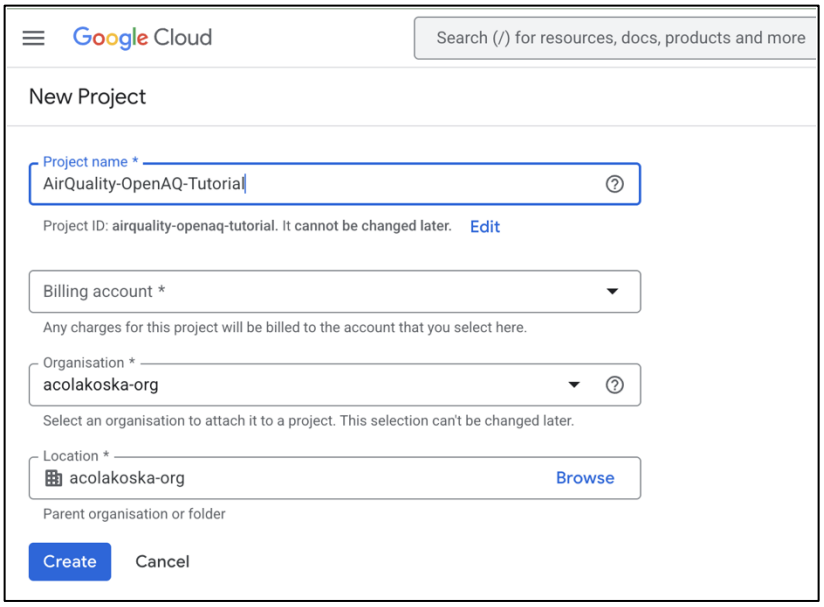

Figure 6.6 Creating a Google Cloud project

To begin, we need to create a dedicated Google Cloud project (Figure 6.6) to keep all resources for the air-quality exercise organized and separate from other work. In the Google Cloud Console, use the project selector in the top navigation bar (it displays the currently active project name) and choose New Project. Assign a clear, descriptive name—such as AirQuality-OpenAQ-Tutorial—so it is easy to recognize later when switching between projects, managing permissions,

or reviewing billing activity. After entering the project name (and selecting an organization or location if prompted), click Create to initialize the project workspace in which all subsequent BigQuery datasets, queries, and dashboards will be managed.

Next, we need to access BigQuery from within the Google Cloud Console. Use the Console search bar to type “BigQuery” and open the BigQuery service. Once it loads, verify that you are working inside the correct project by checking the project selector in the top navigation bar; this ensures that any datasets you create, queries you run, and views you save are associated with the intended project.

Ensure the BigQuery API is enabled (Figure 6.7). In most cases, the BigQuery API is enabled automatically the first time you open BigQuery. However, if you encounter an access or API-related error, you should enable it manually. Navigate to APIs & Services → Enabled APIs & services and confirm that BigQuery API is listed. If it is not enabled, select it and click Enable so the project can execute BigQuery operations such as querying datasets and creating views.

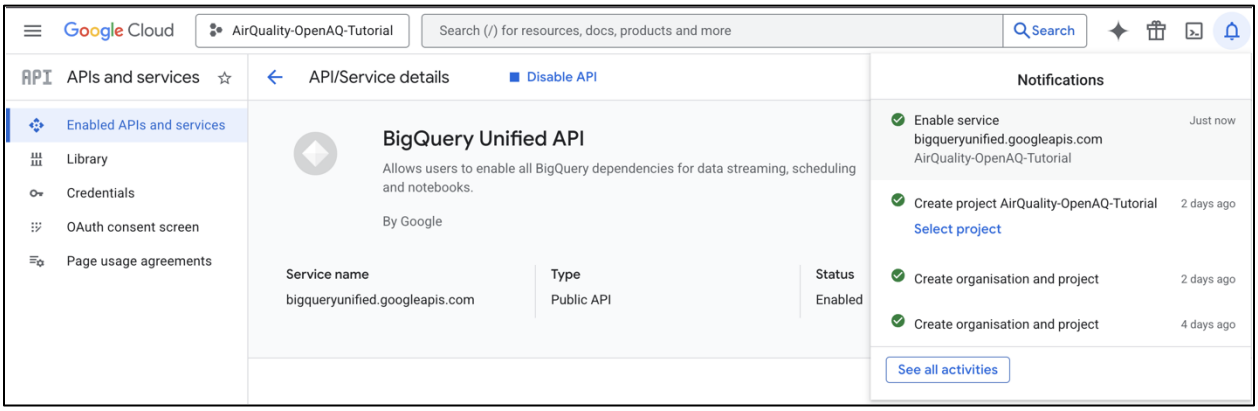

Figure 6.7 Enabling the BigQuery unified API

After this is enabled, we need to add the OpenAQ dataset. The fastest approach to do this is to use the Google Cloud Marketplace. Click Add Data and choose Public datasets, which will open the Marketplace. Then, search for OpenAQ, select the listing, and choose View dataset. This action automatically makes the dataset available in our current Google Cloud project without requiring data movement or storage charges (Figure 6.8).

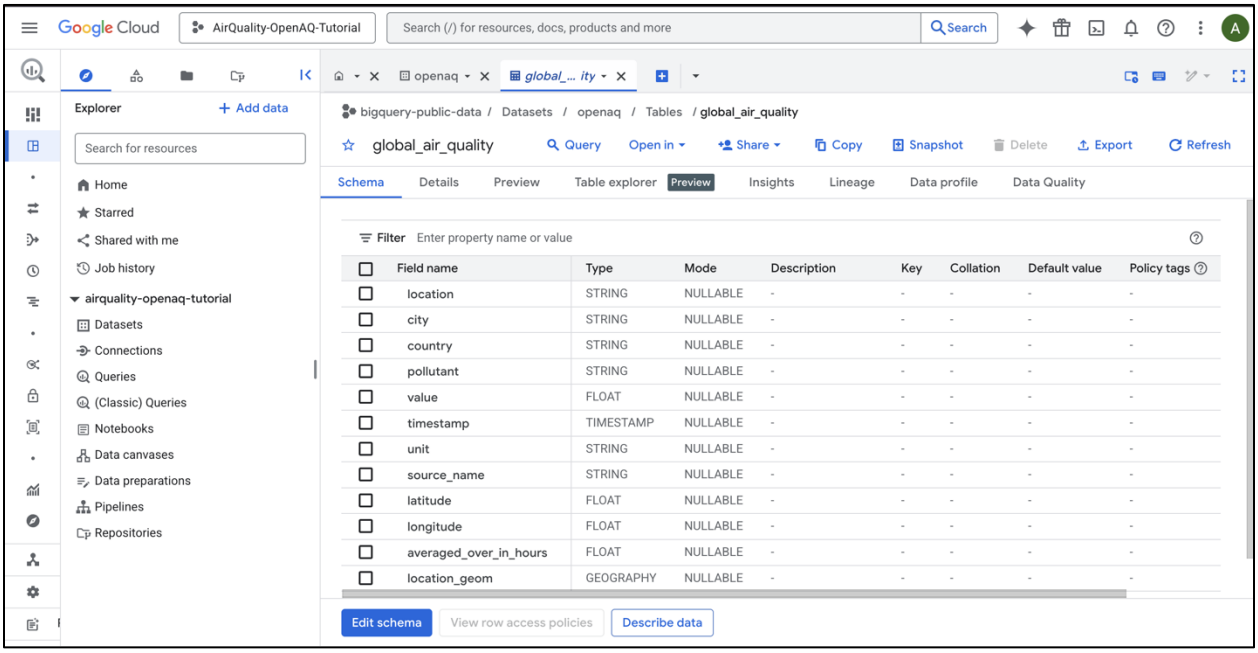

Figure 6.8 Exploring the dataset in Google BigQuery

In the central panel, the Schema view is selected, displaying a structured list of fields (columns) for the global_air_quality table. The schema table includes column headers like Field name, Type, and Mode, and it lists air-quality related attributes such as location, city, country, pollutant, value, and timestamp, along with measurement context fields like unit and source_name. It also includes geographic information such as latitude, longitude, and a spatial field location_geom of type GEOGRAPHY, which suggests the dataset supports mapping and spatial analysis in addition to standard time-series queries.

To analyze OpenAQ air-quality data in a reproducible and dashboard-ready manner, we need to develop several BigQuery queries and store them as views. The first query will aggregate PM2.5 measurements into a daily time series, enabling longitudinal assessment of pollution dynamics and supporting trend visualization. The second query will identify high-pollution episodes by ranking days according to their average PM2.5 concentration, thereby highlighting extreme events of potential public-health relevance. The third query will compute average PM2.5 levels across monitoring locations, producing a comparative ranking that facilitates the detection of persistent hotspots and spatial variability in exposure. Finally, the fourth query will extract the most recent PM2.5 observation per location (subject to valid geographic coordinates) to generate a map-ready layer for spatial visualization. Because the dataset available in BigQuery is a historical snapshot (with a maximum timestamp in 2022), the queries implement time windows relative to the dataset's latest timestamp, ensuring consistent outputs despite the absence of near-real-time

updates. Collectively, these views provide a structured analytical foundation that can be directly connected to Looker Studio for interactive dashboards and transparent reporting (Figure 6.9).

1) PM2.5 daily trend for the last 90 days relative to max_ts

```
WITH t AS (
 SELECT MAX(timestamp) AS max_ts
 FROM `bigquery-public-data.openaq.global_air_quality`
)
SELECT
 DATE(g.timestamp) AS day,
 AVG(g.value) AS avg_pm25
FROM `bigquery-public-data.openaq.global_air_quality` g
CROSS JOIN t
WHERE g.pollutant = 'pm25'
 AND g.timestamp >= TIMESTAMP_SUB(t.max_ts, INTERVAL 90 DAY)
GROUP BY day
ORDER BY day;
```

Save as view: Run → SAVE → Save view → dataset openaq_dashboard → name v_pm25_daily_trend

2) Worst 10 days (highest PM2.5) in the last year relative to max_ts

```
WITH t AS (
 SELECT MAX(timestamp) AS max_ts
 FROM `bigquery-public-data.openaq.global_air_quality`
)
SELECT
 DATE(g.timestamp) AS day,
 AVG(g.value) AS avg_pm25,
 COUNT(*) AS measurements
FROM `bigquery-public-data.openaq.global_air_quality` g
CROSS JOIN t
WHERE g.pollutant = 'pm25'
 AND g.timestamp >= TIMESTAMP_SUB(t.max_ts, INTERVAL 365 DAY)
GROUP BY day
ORDER BY avg_pm25 DESC
LIMIT 10;
```

Save as view: v_pm25_worst_days

3) Top 15 locations by average PM2.5 (last 90 days relative to max_ts)

```
WITH t AS (
  SELECT MAX(timestamp) AS max_ts
  FROM `bigquery-public-data.openaq.global_air_quality`
)
SELECT
  COALESCE(g.country, 'Unknown') AS country,
  COALESCE(g.city, 'Unknown') AS city,
  g.location,
  AVG(g.value) AS avg_pm25,
  COUNT(*) AS measurements
FROM `bigquery-public-data.openaq.global_air_quality` g
CROSS JOIN t
WHERE g.pollutant = 'pm25'
  AND g.timestamp >= TIMESTAMP_SUB(t.max_ts, INTERVAL 90 DAY)
GROUP BY country, city, location
HAVING measurements >= 50
ORDER BY avg_pm25 DESC
LIMIT 15;
```

Save as view: v_pm25_top_locations

4) Latest PM2.5 per location (within last 14 days relative to max_ts)

```
WITH t AS (
  SELECT MAX(timestamp) AS max_ts
  FROM `bigquery-public-data.openaq.global_air_quality`
),
latest AS (
  SELECT
    g.location,
    g.city,
    g.country,
    g.latitude,
```

```
    g.longitude,
    g.value,
    g.unit,
    g.timestamp,
    ROW_NUMBER() OVER (PARTITION BY g.location ORDER BY g.timestamp DESC) AS rn
  FROM `bigquery-public-data.openaq.global_air_quality` g
  CROSS JOIN t
  WHERE g.pollutant = 'pm25'
    AND g.latitude IS NOT NULL
    AND g.longitude IS NOT NULL
    AND g.timestamp >= TIMESTAMP_SUB(t.max_ts, INTERVAL 14 DAY)
)
SELECT
  location,
  city,
  country,
  latitude,
  longitude,
  value AS latest_pm25,
  unit,
  timestamp AS latest_time
FROM latest
WHERE rn = 1;
```

We can now see the results (views) in Figure 6.9.

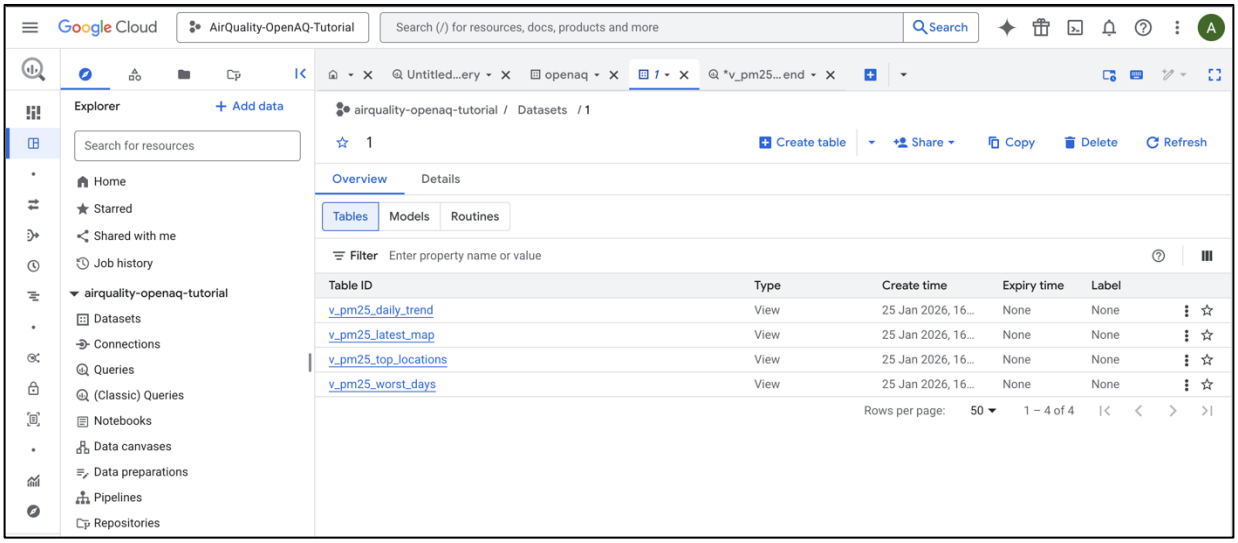

Figure 6.9 Four views – result of the SQL querries

Next, we will use Looker Studio to develop an interactive air-quality dashboard by connecting directly to four precomputed BigQuery views derived from the OpenAQ dataset. By treating each view as a reusable analytical component—covering a daily PM2.5 trend, extreme (“worst day”) events, location-based rankings, and a map-ready layer of the latest observations—the workflow separates data preparation from visualization, improving transparency and reproducibility. The result is a cohesive dashboard that supports exploratory analysis through charts, tables, and geographic mapping, and can be shared for reporting and communication of environmental insights.

To begin, create a new dashboard by opening Looker Studio and selecting Create → Report. If prompted, authenticate with your Google account and grant the necessary permissions so Looker Studio can access your Google Cloud resources. Once the blank report opens, connect your first dataset by clicking Add data and selecting BigQuery as the connector. From the BigQuery navigator, choose the correct Project (e.g., AirQuality-OpenAQ-Tutorial), then select the dataset openaq_dashboard and the view v_pm25_daily_trend. Finally, click Add (and confirm Add to report if requested) to attach the daily trend view as the initial data source for your dashboard (Figure 6.10).

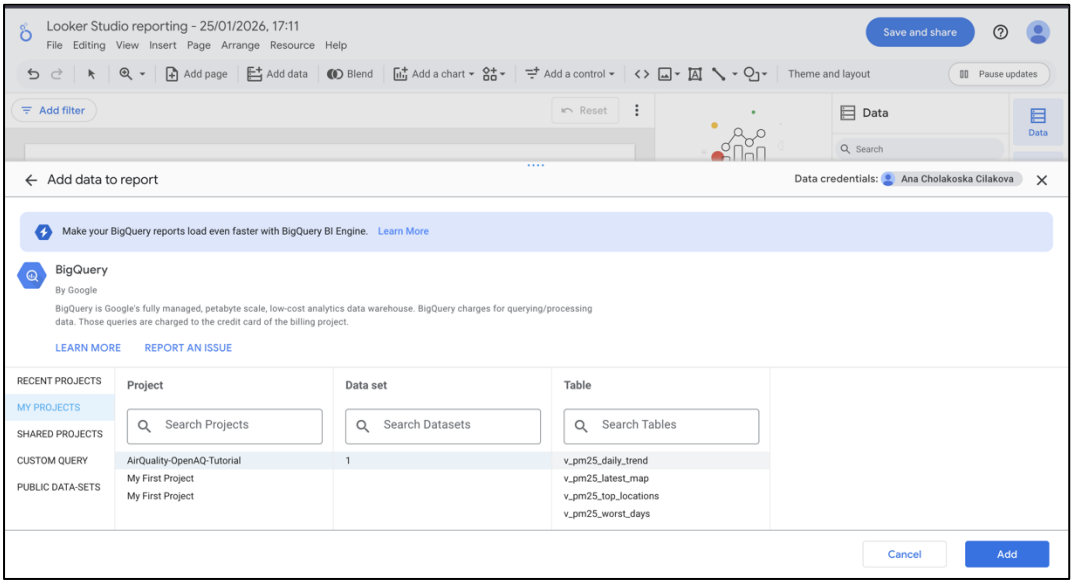

Figure 6.10 Adding data to report

After adding the v_pm25_daily_trend view as a data source, we need to create a Time Series chart to represent temporal variation in air pollution. Insert the chart onto the report canvas and configure the date field (day) as the primary dimension and average PM2.5 (avg_pm25) as the metric. This visualization provides an interpretable summary of day-to-day fluctuations and longer-term tendencies in particulate concentrations, supporting the identification of sustained increases, decreases, or episodic peaks that may correspond to meteorological conditions or emission events.

Using the v_pm25_top_locations view, add a Bar Chart to compare average PM2.5 concentrations across monitoring sites. Place the chart in the report and set location (or city, depending on the desired aggregation) as the dimension and avg_pm25 as the metric, sorting the bars in descending order to emphasize the highest-exposure locations. This chart enables rapid hotspot recognition by ranking sites according to their mean concentration over the analysis window and can be complemented by the measurements field to contextualize the robustness of each estimate.

To highlight extreme pollution events, insert a Table visualization based on the v_pm25_worst_days view. Configure the table with day as the dimension and include avg_pm25 (and, where available, measurements) as metrics, sorting by average PM2.5 in descending order to keep the most severe days at the top. This tabular representation supports precise reporting

by displaying exact values and facilitating comparisons between peak days, which is particularly useful for summarizing environmental risk periods in a concise, audit-friendly format.

Finally, to support spatial interpretation, create a Google Maps chart using the v_pm25_latest_map view. Add the map to the canvas and assign latitude and longitude as the geographic fields, with latest_pm25 as the key metric to encode the magnitude of recent observations (e.g., through marker size or intensity). Enhancing tooltips with contextual attributes such as location, city, and latest_time enables users to inspect individual monitoring points, thereby combining geographic context with quantitative values to communicate where elevated concentrations are most recently observed. We can see all the graphs in Figure 6.11.

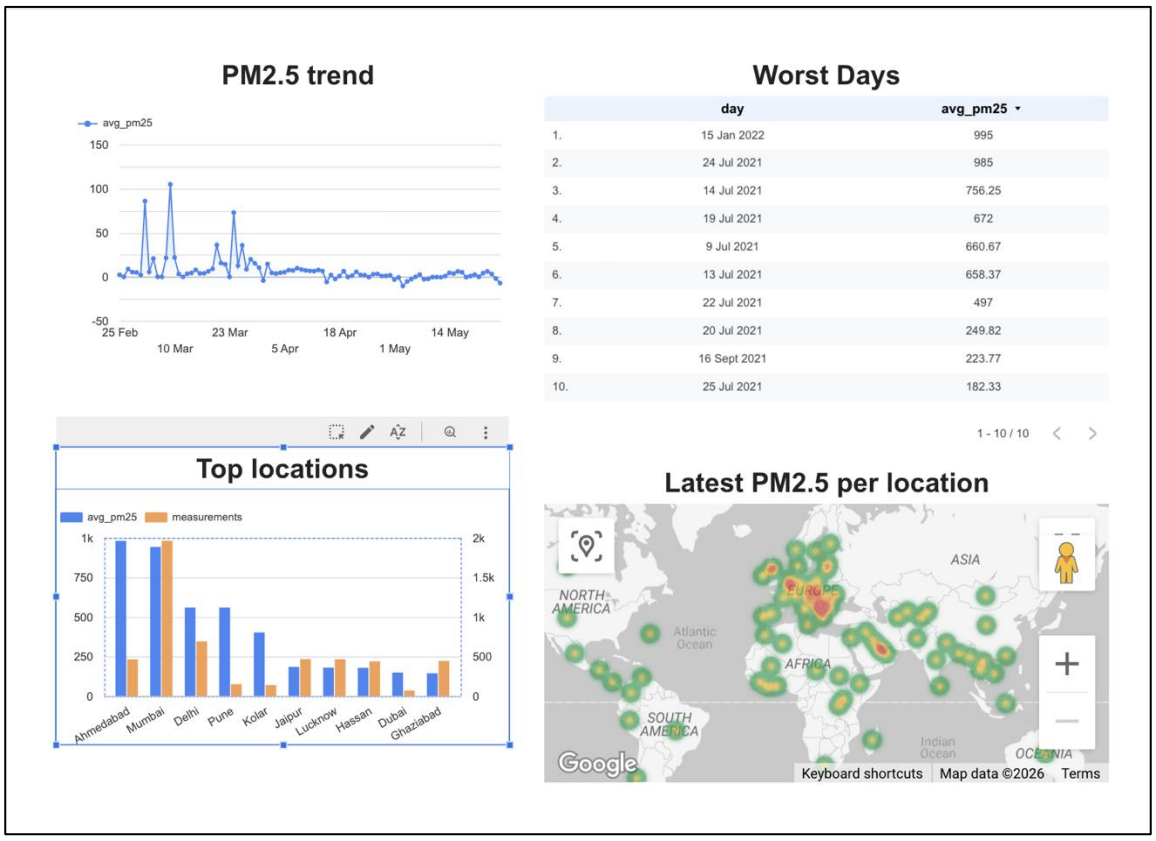

Figure 6.11 The complete dashboard

Optionally, we can proceed to optimizing the fourth query to reduce computational overhead and improve cost efficiency while preserving identical analytical intent (i.e., retrieving the most recent PM2.5 measurement per monitoring location). First, the common-table expression and post-filter (WHERE rn = 1) are replaced with a QUALIFY clause, enabling BigQuery to apply the window-function filter directly within the same scan, which simplifies execution and typically improves performance. Second, explicit timestamp IS NOT NULL checks are added to prevent unnecessary processing of invalid records and to ensure the time filter behaves consistently. Third, the original "last 14 days from the current date" condition is reformulated to use a window relative to the dataset's maximum available timestamp (MAX(timestamp)), which is essential for historical snapshots (such as OpenAQ tables that are not continuously updated) and guarantees non-empty outputs. Finally, by selecting only the fields required for mapping and reporting (location metadata, coordinates, value, unit, and time), the query minimizes column reads, supporting lower bytes processed and faster dashboard refreshes. Below we can see the full code snippet:

```
SELECT
 location,
 city,
 country,
 latitude,
 longitude,
 value AS latest_pm25,
 unit,
 timestamp AS latest_time
FROM `bigquery-public-data.openaq.global_air_quality`
WHERE pollutant = 'pm25'
 AND timestamp IS NOT NULL
 AND latitude IS NOT NULL
 AND longitude IS NOT NULL
 -- Use dataset's latest timestamp window if your snapshot is old:
 AND timestamp >= TIMESTAMP_SUB(
```

```
  (SELECT MAX(timestamp) FROM `bigquery-public-data.openaq.global_air_quality` WHERE timestamp IS NOT NULL),
  INTERVAL 14 DAY
 )
QUALIFY ROW_NUMBER() OVER (PARTITION BY location ORDER BY timestamp DESC) = 1;
```

Practical Exercise

Mini Project: Select an environmental dataset (e.g., air quality, marine, or biodiversity). Clean, process, query, and visualize results using Google Cloud tools. Present workflow and findings in a short presentation (5–10 minutes).

Assessment

1. Which option best matches the 3Vs of Big Data?

A) Volume = data accuracy, Velocity = data size, Variety = data speed

B) Volume = amount of data, Velocity = speed of data generation/processing, Variety = different data types/sources

C) Volume = number of users, Velocity = internet speed, Variety = number of servers

D) Volume = storage cost, Velocity = query cost, Variety = number of dashboards

2. Which pairing is most accurate?

A) CSV = unstructured, JSON = structured, images = semi-structured

B) CSV = structured, JSON = semi-structured, images/videos = unstructured

C) CSV = semi-structured, JSON = unstructured, images = structured

D) CSV = unstructured, JSON = unstructured, images = structured

3. You need fast analytics on large event logs (columns like timestamp, device_id, PM2.5, temperature) stored in cloud object storage. Best choice?

A) Parquet/ORC (columnar)

B) Plain text (.txt)

C) PNG images

D) XML with deeply nested tags

4. Which set correctly matches the service to the task?

A) BigQuery = object storage, Cloud Storage = SQL analytics, Dataflow = visualization

B) Cloud Storage = object storage, BigQuery = serverless analytics warehouse, Dataflow = distributed data processing (ETL)

C) BigQuery = message queue, Pub/Sub = dashboards, Looker = data ingestion

D) Dataflow = file storage, Cloud Storage = ML training, BigQuery = identity management

5. Which action usually reduces query cost and runtime in a cloud data warehouse (e.g., BigQuery) for time-series data?

A) Use SELECT * to avoid missing columns

B) Partition by date/time and cluster by common filter keys (e.g., station_id/device_id)

C) Store everything as one giant JSON column to simplify schema

D) Disable caching and rerun queries repeatedly to “warm up” the system

References / Further Reading

[1] BigQuery public datasets (overview + how to access). Available: https://docs.cloud.google.com/bigquery/public-data, Accessed: Jan. 15, 2026

[2] BigQuery: Introduction to optimizing query performance. Available: https://docs.cloud.google.com/bigquery/docs/best-practices-performance-overview, Accessed: Jan. 15, 2026

[3] BigQuery: Optimize query computation (best practices). Available: https://docs.cloud.google.com/bigquery/docs/best-practices-performance-compute, Accessed: Jan. 15, 2026

[4] Looker Studio: Connect to Google BigQuery. Available: https://docs.cloud.google.com/looker/docs/studio/connect-to-google-bigquery, Accessed: Jan. 15, 2026

[5] Looker Studio Documentation. Available: https://docs.cloud.google.com/looker/docs/studio, Accessed: Jan. 15, 2026

[6] Looker Studio quick start guide . Available: https://docs.cloud.google.com/looker/docs/studio/quick-start-guide, Accessed: Jan. 15, 2026

[7] Cloud Billing: Budgets and budget alerts. Available: https://docs.cloud.google.com/billing/docs/how-to/budgets, Accessed: Jan. 15, 2026

[8] Cloud Billing documentation hub. Available: https://docs.cloud.google.com/billing/docs, Accessed: Jan. 15, 2026

[9] BigQuery sharing (formerly Analytics Hub): Introduction. Available: https://docs.cloud.google.com/bigquery/docs/analytics-hub-introduction, Accessed: Jan. 15, 2026

[10] BigQuery Explained (architecture overview, Google Cloud Blog). Available: https://cloud.google.com/blog/products/data-analytics/new-blog-series-bigquery-explained-overview, Accessed: Jan. 15, 2026