Independent Skill Development Mission

Independent Skill Development NXT Mission
EMPOWERING FUTURES, SHAPING TOMORROW

ISDM (INDEPENDENT SKILL DEVELOPMENT MISSION

BIGQUERY – ANALYZING LARGE DATASETS

CHAPTER 1: INTRODUCTION TO BIGQUERY

1.1 What is BigQuery?

BigQuery is a **fully managed, serverless, and highly scalable data warehouse** provided by **Google Cloud Platform (GCP)** for **analyzing large datasets** in real-time. It allows users to execute **SQL-like queries** on petabyte-scale datasets efficiently.

1.2 Key Features of BigQuery

✓ **Serverless** – No infrastructure management required.
✓ **Fast SQL Queries** – Uses **Dremel technology** for quick query execution.
✓ **Built-in Machine Learning (BigQuery ML)** – Enables **ML model training** using SQL.
✓ **Streaming Data Ingestion** – Supports **real-time analytics**.
✓ **Security & Compliance** – Supports **IAM-based access control** and encryption.

📌 **Example:**
An **e-commerce company** uses **BigQuery** to analyze **customer purchasing behavior** and improve product recommendations.

CHAPTER 2: BIGQUERY ARCHITECTURE

2.1 How BigQuery Works?

BigQuery follows a **distributed architecture** with the following components:

Component	Description
Storage Engine	Stores **structured** and **semi-structured** data efficiently.
Query Engine	Uses **Dremel execution** for **fast SQL queries**.
BigQuery ML	Enables **machine learning model training** within BigQuery.
BigQuery BI Engine	Optimizes **dashboard performance** in **Looker, Data Studio, etc.**
Data Transfer Service	Automates data movement from **Google Ads, YouTube, and other sources**.

📌 **Example:**

A **marketing team** imports **Google Analytics data** into BigQuery for real-time customer insights.

CHAPTER 3: SETTING UP BIGQUERY

3.1 Prerequisites

✓ **Google Cloud Project** with **billing enabled**.

✓ Enable the **BigQuery API**.

✓ Install **Google Cloud SDK** (for CLI access).

3.2 Creating a BigQuery Dataset

Step 1: Open BigQuery Console

1. Navigate to **Google Cloud Console** → **BigQuery**.
2. Click **Create Dataset** → Name it my_dataset.

Step 2: Upload Data to BigQuery

1. Open the **BigQuery Console**.
2. Click **Create Table** → Choose **Upload File**.
3. Select **CSV, JSON, or Avro file** → Click **Create Table**.

📌 **Example:**
A **finance company** uploads **transaction logs** to BigQuery for **fraud detection analysis**.

Chapter 4: Querying Data in BigQuery

4.1 Running Basic SQL Queries

BigQuery supports **SQL-like syntax** to retrieve data.

Example: Selecting Data

```
SELECT customer_id, total_spent

FROM `my_project.my_dataset.sales_data`

WHERE total_spent > 1000

ORDER BY total_spent DESC

LIMIT 10;
```

Example: Aggregating Data

```
SELECT category, COUNT(*) AS num_orders

FROM `my_project.my_dataset.orders`

GROUP BY category
```

```
ORDER BY num_orders DESC;
```

📌 **Example:**
An **online retailer** queries BigQuery to identify the **top-selling product categories**.

Chapter 5: Optimizing Query Performance

5.1 Best Practices for Faster Queries

✓ **Partition Tables** – Store data in **time-based partitions**.
✓ **Use Clustering** – Group frequently queried fields together.
✓ **Avoid SELECT *** – Query only required columns.
✓ **Use Query Caching** – Reduce cost and execution time.

Example: Using Partitioning in BigQuery

```
CREATE TABLE my_project.my_dataset.sales_data

PARTITION BY DATE(transaction_date)

AS

SELECT * FROM `source_table`;
```

📌 **Example:**
A **logistics company** improves **shipment tracking performance** using **partitioned tables**.

Chapter 6: BigQuery ML – Machine Learning in SQL

6.1 What is BigQuery ML?

BigQuery ML enables users to **train machine learning models using SQL queries**.

6.2 Creating an ML Model in BigQuery

```
CREATE MODEL my_project.my_dataset.customer_churn_model
OPTIONS(model_type='logistic_reg') AS
SELECT * FROM my_project.my_dataset.customer_data;
```

📌 **Example:**
A **telecom company** predicts **customer churn** using **BigQuery ML**.

Chapter 7: Security & Access Control in BigQuery

7.1 Implementing IAM Policies

✔ **Role-Based Access Control (RBAC)** – Assign roles such as **Viewer, Editor, Owner**.
✔ **Row-Level Security** – Restrict access at **row-level**.
✔ **Data Encryption** – Supports **Customer-Managed Encryption Keys (CMEK)**.

📌 **Example:**
A **healthcare organization** ensures **HIPAA compliance** by **restricting patient data access**.

Chapter 8: Exercise & Review Questions

Exercise:

1. **Upload a dataset** to BigQuery and create a table.
2. **Run SQL queries** to filter and aggregate data.
3. **Optimize performance** using partitions and clustering.
4. **Train an ML model** using BigQuery ML.

Review Questions:

1. What are the key components of **BigQuery Architecture**?
2. How does **BigQuery ML** enable machine learning?
3. What is the difference between **partitioning and clustering**?
4. What are the **best practices for optimizing BigQuery queries**?
5. How can **IAM roles** enhance security in BigQuery?

Conclusion: Unlocking the Power of BigQuery

✓ **BigQuery simplifies large-scale data analytics** with SQL-based querying.
✓ **Optimizations like partitioning & clustering** improve query performance.
✓ **Built-in ML capabilities** enable predictive analytics.

🚀 **Mastering BigQuery helps businesses make data-driven decisions efficiently!**

GOOGLE CLOUD DATAFLOW – STREAM & BATCH PROCESSING PIPELINES

CHAPTER 1: INTRODUCTION TO GOOGLE CLOUD DATAFLOW

1.1 What is Google Cloud Dataflow?

Google Cloud Dataflow is a **fully managed service** for **stream and batch data processing**, built on **Apache Beam**. It allows developers to **process, analyze, and transform large datasets** in **real-time (stream processing)** or as **scheduled jobs (batch processing)**.

1.2 Key Features of Dataflow

✓ **Serverless Execution** – No need to manage infrastructure.
✓ **Unified Model** – Supports both **batch and stream processing**.
✓ **Auto-Scaling** – Dynamically scales resources based on demand.
✓ **Pay-for-Use Pricing** – Charges only for the compute resources used.
✓ **Integration with BigQuery, Pub/Sub, Cloud Storage, and AI tools**.

📌 **Example:**
An **e-commerce company** uses **Dataflow with Pub/Sub** to **process real-time sales transactions** and generate **real-time analytics dashboards**.

CHAPTER 2: UNDERSTANDING BATCH & STREAM PROCESSING

2.1 What is Batch Processing?

Batch processing **handles large volumes of data at once**, often at **scheduled intervals**.

✓ Use Cases:

- Processing **log files** from a website.
- Performing **ETL (Extract, Transform, Load) jobs**.
- Generating **daily reports** from stored data.

✓ Example:
A **finance company** processes **daily stock market transactions** using **Dataflow batch processing** to calculate **end-of-day summaries**.

2.2 What is Stream Processing?

Stream processing **processes data in real time** as it arrives.

✓ Use Cases:

- **Real-time fraud detection** for banking transactions.
- **Monitoring IoT sensor data**.
- **Processing live customer interactions** (e.g., chatbots, recommendations).

✓ Example:
A **media streaming service** uses **Dataflow to analyze live video viewership trends** and **adjust recommendations dynamically**.

Chapter 3: Architecture of Google Cloud Dataflow

3.1 Core Components of Dataflow

Component	Description
Pipeline	The flow of data from input to output.

Transform	Operations applied to data (filtering, aggregating, etc.).
PCollection	A distributed dataset in Dataflow.
Source	The input data source (e.g., Pub/Sub, Cloud Storage, BigQuery).
Sink	The output destination (e.g., BigQuery, Cloud Storage, Datastore).

📌 **Example:**

A **retail company** uses Dataflow to process **customer transaction logs** from **Cloud Storage** and load the **results into BigQuery**.

3.2 Dataflow vs Other Google Cloud Services

Feature	**Dataflow**	**BigQuery**	**Dataproc**
Processing Model	Batch & Stream	Query-based (SQL)	Managed Hadoop/Spark
Scalability	Auto-scales	Auto-scales	Cluster-based
Use Case	Real-time & batch processing	Data warehousing	Big data analytics
Best For	ETL, real-time analytics	SQL analytics	Custom ML & big data jobs

📌 **Example:**

A **telecom provider** uses **Dataflow** to **process call records** in **real-time**, while storing historical data in **BigQuery**.

CHAPTER 4: SETTING UP A DATAFLOW PIPELINE

4.1 Prerequisites

✓ Enable the **Dataflow API** in **Google Cloud Console**.
✓ Install **Google Cloud SDK** (gcloud init).
✓ Install **Apache Beam SDK**:

```
pip install apache-beam[gcp]
```

4.2 Creating a Dataflow Pipeline Using Python (Apache Beam)

Step 1: Define the Pipeline

```
import apache_beam as beam

from apache_beam.options.pipeline_options import PipelineOptions



pipeline_options = PipelineOptions()

p = beam.Pipeline(options=pipeline_options)



# Read data from Cloud Storage

input_data = 'gs://my-bucket/input-data.csv'



lines = p | "Read from Cloud Storage" >> beam.io.ReadFromText(input_data)



# Apply transformation (e.g., filtering data)

filtered_lines = lines | "Filter" >> beam.Filter(lambda line: "error" in line.lower())
```

```
# Write output to Cloud Storage

output_data = 'gs://my-bucket/output-data.txt'

filtered_lines | "Write to Cloud Storage" >> beam.io.WriteToText(output_data)


p.run()
```

Step 2: Deploy the Pipeline on Dataflow

```
python my_pipeline.py --runner DataflowRunner --project my-gcp-project --temp_location gs://my-bucket/temp/
```

📌 **Example:**
A **log management system** filters **error logs** from **Cloud Storage** and writes **filtered logs back to Cloud Storage**.

CHAPTER 5: INTEGRATING DATAFLOW WITH OTHER GOOGLE CLOUD SERVICES

5.1 Dataflow with Pub/Sub for Real-Time Processing

1. **Pub/Sub receives streaming data** (e.g., IoT, transactions).
2. **Dataflow processes the stream** (e.g., filtering, aggregating).
3. **Data is written to BigQuery for analysis**.

✓ **Use Case:**

- **Fraud detection in financial transactions**.
- **Monitoring real-time customer orders**.

Sample Code: Streaming Data from Pub/Sub to BigQuery

```
from apache_beam.io.gcp.pubsub import ReadFromPubSub

from apache_beam.io.gcp.bigquery import WriteToBigQuery


p = beam.Pipeline()


messages = (

   p

   | "Read from Pub/Sub" >> ReadFromPubSub(subscription="projects/my-project/subscriptions/my-subscription")

   | "Transform Data" >> beam.Map(lambda msg: {"message": msg.decode("utf-8")})

   | "Write to BigQuery" >> WriteToBigQuery("my-project:dataset.table")

)


p.run()
```

📌 **Example:**
A **cybersecurity firm** uses **Dataflow with Pub/Sub** to **detect unauthorized access attempts in real time**.

Chapter 6: Best Practices for Dataflow Pipelines

✔ **Use Windowing for Streaming Data** – Helps group real-time data into time-based windows.

✔ **Optimize Resource Allocation** – Use **auto-scaling** to control costs.

✓ **Use Dead-Letter Queues** – Redirect failed records to **Cloud Storage or Pub/Sub**.

✓ **Enable Logging & Monitoring** – Use **Cloud Logging & Stackdriver** for debugging.

✓ **Secure Dataflow Pipelines** – Implement **IAM roles** to restrict access.

📌 **Example:**

A **healthcare provider** ensures **data security** by using **IAM roles** to restrict access to **Dataflow pipelines processing patient records**.

Chapter 7: Exercise & Review Questions

Exercise:

1. **Deploy a batch pipeline** that processes CSV files from Cloud Storage.
2. **Create a streaming pipeline** that ingests data from Pub/Sub and writes to BigQuery.
3. **Optimize a Dataflow job** by setting auto-scaling and monitoring logs.

Review Questions:

1. What is the difference between **batch processing and stream processing**?
2. How does **Apache Beam** enable portability in Dataflow pipelines?
3. What are **PCollections** in Dataflow?
4. How can **windowing** improve real-time data processing?

5. What is the role of **Pub/Sub in streaming pipelines**?

Conclusion: Leveraging Dataflow for Scalable Data Processing

✔ **Google Cloud Dataflow enables real-time and batch data processing at scale.**
✔ **Integration with Pub/Sub, BigQuery, and Cloud Storage** makes it ideal for ETL workflows.
✔ **Serverless execution, auto-scaling, and monitoring features** optimize performance.

🚀 **Mastering Dataflow helps businesses process, analyze, and transform data efficiently in Google Cloud!**

CLOUD DATAPROC – RUNNING SPARK & HADOOP ON GCP

CHAPTER 1: INTRODUCTION TO CLOUD DATAPROC

1.1 What is Cloud Dataproc?

Cloud Dataproc is a **fully managed, cloud-native service** provided by Google Cloud for running **Apache Hadoop, Apache Spark, Apache Flink, Presto, and other big data frameworks**. It enables organizations to process large-scale data workloads **cost-effectively and efficiently**.

1.2 Why Use Cloud Dataproc?

✓ **Fully Managed** – No need to set up or manage infrastructure.

✓ **Fast Cluster Deployment** – Clusters can be created in **under 90 seconds**.

✓ **Scalable** – Auto-scaling allows dynamic resource allocation.

✓ **Cost-Effective** – Pay only for what you use with per-second billing.

✓ **Integrated with GCP Services** – Works with BigQuery, Cloud Storage, Cloud Dataflow, and AI/ML services.

1.3 Key Use Cases

✓ **Data Transformation & ETL Pipelines** – Process structured and unstructured data.

✓ **Machine Learning & AI Pipelines** – Train models using big data frameworks.

✓ **Real-time & Batch Data Processing** – Process streaming or batch workloads.

✓ **Ad-hoc Querying & Data Exploration** – Run Spark SQL queries on large datasets.

📌 **Example:**
A retail company uses **Cloud Dataproc to analyze customer purchase trends** by processing large datasets stored in **Google Cloud Storage** using **Apache Spark**.

Chapter 2: Setting Up Cloud Dataproc Cluster

2.1 Prerequisites

✓ **Google Cloud Project** with billing enabled.
✓ **Enable Cloud Dataproc API**:

```
gcloud services enable dataproc.googleapis.com
```

✓ **Set Up IAM Permissions** for managing Dataproc clusters.
✓ **Install Google Cloud SDK** and authenticate using:

```
gcloud auth login
```

2.2 Creating a Dataproc Cluster

Method 1: Using Google Cloud Console

1. Open **Google Cloud Console** → **Cloud Dataproc**.
2. Click **Create Cluster**.
3. Set the **Cluster Name** (e.g., my-dataproc-cluster).
4. Choose the **Region** and **Zone**.
5. Select **Cluster Mode**:
   - **Standard** – Multi-node cluster (Default).
   - **Single Node** – Best for testing.

    - **High Availability** – Multiple master nodes for reliability.

6. Configure **Machine Types & Nodes**.
7. Click **Create** to launch the cluster.

Method 2: Using Google Cloud CLI

```
gcloud dataproc clusters create my-dataproc-cluster \
  --region=us-central1 \
  --zone=us-central1-a \
  --master-machine-type=n1-standard-4 \
  --worker-machine-type=n1-standard-2 \
  --num-workers=2
```

📌 **Example:**
A **finance company** sets up a **Dataproc cluster with two worker nodes** to process large financial transactions in real time.

Chapter 3: Running Spark & Hadoop Jobs on Dataproc

3.1 Running an Apache Spark Job

Submit a Spark Job Using the Cloud Console

1. Open **Cloud Dataproc** → **Jobs**.
2. Click **Submit Job** → Select **Spark** as the Job Type.
3. Specify the main application file (e.g., a JAR file stored in **Cloud Storage**).
4. Provide necessary **arguments & parameters**.
5. Click **Submit**.

Submit a Spark Job Using CLI

```
gcloud dataproc jobs submit spark \
  --cluster=my-dataproc-cluster \
  --region=us-central1 \
  --class=org.apache.spark.examples.SparkPi \
  --jars=file:///usr/lib/spark/examples/jars/spark-examples.jar \
  -- 1000
```

📌 **Example:**
A **social media company** runs a **Spark job** on Dataproc to analyze trending hashtags from millions of tweets.

3.2 Running a Hadoop Job

Submit a Hadoop Job Using CLI

```
gcloud dataproc jobs submit hadoop \
  --cluster=my-dataproc-cluster \
  --region=us-central1 \
  --class=org.apache.hadoop.examples.WordCount \
  --jars=file:///usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar \
  -- input.txt output-dir
```

📌 **Example:**
A **news website** runs a **Hadoop MapReduce job** to count the number of words in an article dataset stored in Cloud Storage.

CHAPTER 4: DATA PROCESSING WITH CLOUD DATAPROC

4.1 Integrating Dataproc with Cloud Storage

Dataproc uses **Cloud Storage (GCS) as a storage backend** instead of HDFS.

```
gcloud storage buckets create my-dataproc-bucket --region=us-central1
```

✓ Upload input data to Cloud Storage:

```
gcloud storage cp local-data.csv gs://my-dataproc-bucket/
```

✓ Read data in Spark using Python (PySpark):

```
df = spark.read.csv("gs://my-dataproc-bucket/local-data.csv", header=True, inferSchema=True)

df.show()
```

📌 **Example:**
An **e-commerce company** stores sales data in **GCS** and processes it using **Apache Spark on Dataproc**.

CHAPTER 5: OPTIMIZING & SCALING CLOUD DATAPROC

5.1 Autoscaling Dataproc Clusters

Enable auto-scaling for cost efficiency:

```
gcloud dataproc clusters update my-dataproc-cluster \

  --region=us-central1 \

  --autoscaling-policy=default
```

5.2 Optimizing Spark Performance on Dataproc

✓ Use **cluster preemptible VMs** to reduce costs.

✓ Enable **dynamic resource allocation** in Spark.

✓ Optimize **shuffle performance using SSD disks**.

📌 **Example:**

A **biotech company** reduces cloud computing costs by enabling **auto-scaling and using preemptible nodes**.

CHAPTER 6: SECURITY & MONITORING IN CLOUD DATAPROC

6.1 Securing Dataproc Clusters

✓ Use **IAM roles** to control access.

✓ Enable **VPC Service Controls** for network security.

✓ Use **Cloud Storage ACLs** to manage data permissions.

6.2 Monitoring Dataproc Jobs

✓ View logs in **Cloud Logging**:

```
gcloud dataproc jobs list --region=us-central1
```

✓ Enable **Cloud Monitoring & Alerting**.

📌 **Example:**

A **banking organization** enables **role-based IAM access** to restrict sensitive data processing.

CHAPTER 7: EXERCISE & REVIEW QUESTIONS

Exercise:

1. **Create a Dataproc Cluster** with two worker nodes.
2. **Run a Spark job** using PySpark.

3. **Process a dataset** stored in Cloud Storage.
4. **Enable Autoscaling** for cost optimization.

Review Questions:

1. What are the **benefits of using Cloud Dataproc over on-premise Hadoop clusters**?
2. How can **Cloud Storage be used instead of HDFS** in Dataproc?
3. What are **the differences between running Spark and Hadoop jobs** on Dataproc?
4. How does **auto-scaling improve cost efficiency** in Cloud Dataproc?
5. What are **best practices for securing Dataproc clusters**?

Conclusion: Cloud Dataproc for Scalable Data Processing

✓ **Google Cloud Dataproc provides an efficient way to run big data frameworks** like Spark & Hadoop.
✓ **Integrates seamlessly with GCS, BigQuery, and AI/ML services**.
✓ **Cost-effective with auto-scaling & per-second billing**.
✓ **Secure, managed, and scalable for modern data processing workflows**.

🚀 **Mastering Cloud Dataproc helps organizations accelerate big data analytics and machine learning in the cloud!**

CLOUD PUB/SUB – REAL-TIME MESSAGING & EVENT PROCESSING

CHAPTER 1: INTRODUCTION TO CLOUD PUB/SUB

1.1 WHAT IS CLOUD PUB/SUB?

Cloud Pub/Sub is a **real-time messaging service** provided by Google Cloud that enables **asynchronous communication** between independent applications. It follows the **publish-subscribe messaging model**, ensuring **scalable, reliable, and event-driven** communication.

✓ **Real-time event processing** – Enables real-time data streaming.

✓ **Asynchronous messaging** – Decouples producers (publishers) and consumers (subscribers).

✓ **Scalable & fault-tolerant** – Handles millions of messages per second.

✓ **Push & Pull delivery** – Supports multiple message consumption models.

1.2 Use Cases of Cloud Pub/Sub

✓ **Data Streaming** – Process IoT device data in real-time.

✓ **Event-driven Systems** – Notify services when new events occur.

✓ **Log Aggregation** – Collect and analyze logs from multiple sources.

✓ **Decoupling Microservices** – Enable communication between microservices.

✓ **Real-time Analytics** – Process and analyze financial transactions, stock trades, etc.

📌 Example:

A **financial services company** uses **Cloud Pub/Sub** to process **stock market data** and update dashboards in real-time.

Chapter 2: Cloud Pub/Sub Architecture

2.1 Key Components of Cloud Pub/Sub

Component	**Description**
Publisher	Produces messages and sends them to a topic.
Topic	A named resource where messages are sent.
Subscription	A connection point that allows consumers to receive messages from a topic.
Subscriber	Applications or services that process messages from a subscription.
Message Acknowledgment	Subscribers acknowledge messages to remove them from the queue.

📌 Example:

A **ride-sharing app** uses **Cloud Pub/Sub** to **update ride statuses**, ensuring drivers and riders get **real-time updates**.

Chapter 3: Setting Up Cloud Pub/Sub

3.1 Prerequisites

✔ A **Google Cloud account** with billing enabled.

✔ Install **Google Cloud SDK** and authenticate using gcloud auth login.

✓ Enable the **Pub/Sub API**:

gcloud services enable pubsub.googleapis.com

3.2 Creating a Pub/Sub Topic

1. Open **Google Cloud Console** → Navigate to **Pub/Sub**.
2. Click **Create Topic** → Enter my-topic.
3. Choose **No message retention** (default) or enable message retention.
4. Click **Create**.

📌 **Example:**

A **weather monitoring system** creates a topic called weather-updates where IoT sensors publish temperature readings.

Chapter 4: Publishing & Subscribing to Messages

4.1 Creating a Subscription

1. Open **Google Cloud Console** → Go to **Pub/Sub** → **Subscriptions**.
2. Click **Create Subscription** → Select my-topic.
3. Choose **Delivery Type**:
    - **Pull** – Subscribers fetch messages manually.
    - **Push** – Messages are automatically pushed to an HTTP endpoint.
4. Click **Create**.

4.2 Publishing a Message

Using **gcloud CLI**:

```
gcloud pubsub topics publish my-topic --message "Hello, Pub/Sub!"
```

4.3 Pulling Messages from a Subscription

```
gcloud pubsub subscriptions pull my-subscription --auto-ack
```

📌 **Example:**

An **e-commerce platform** publishes an event when a new **order is placed**, and a **warehouse system subscribes** to process orders in real-time.

CHAPTER 5: CLOUD PUB/SUB MESSAGE DELIVERY MODELS

5.1 Push vs Pull Subscriptions

Feature	**Push Subscription**	**Pull Subscription**
Delivery	Google Cloud pushes messages to a subscriber's HTTP endpoint	Subscriber pulls messages manually
Use Case	Real-time processing (e.g., webhooks)	Batch processing (e.g., analytics, log aggregation)
Latency	Lower latency	Higher latency

5.2 Dead Letter Topics (DLT)

✔ Helps **store undeliverable messages** for debugging.

✔ Messages exceeding the **maximum delivery attempts** are sent to a **dead-letter topic**.

📌 **Example:**

A **chat application** uses a **pull subscription** to process **user messages asynchronously**.

CHAPTER 6: SECURING & MONITORING PUB/SUB

6.1 Security Best Practices

✓ Use **IAM roles** (roles/pubsub.publisher & roles/pubsub.subscriber).

✓ Enable **encryption** for sensitive messages.

✓ Implement **access control policies** for topics and subscriptions.

✓ Use **Google Cloud Audit Logs** to track message activity.

6.2 Monitoring with Cloud Logging & Metrics

✓ View **message delivery statistics** in **Cloud Monitoring**.

✓ Use **Cloud Logging** to track failed deliveries.

✓ Set up **alerts** for failed subscriptions.

📌 **Example:**

A **healthcare system** encrypts Pub/Sub messages to **secure patient records** shared across hospitals.

CHAPTER 7: INTEGRATING CLOUD PUB/SUB WITH OTHER SERVICES

Google Cloud Service	**Integration Purpose**
Cloud Functions	Trigger functions on new Pub/Sub messages.
Cloud Run	Process messages with serverless applications.
BigQuery	Ingest and analyze streaming data.
Cloud Dataflow	Transform and process messages at scale.

📌 **Example:**
A **smart home automation system** integrates Pub/Sub with **Cloud Functions** to **trigger alarms** when motion is detected.

Chapter 8: Exercise & Review Questions

Exercise:

1. **Create a Pub/Sub topic** and publish a message.
2. **Set up a push subscription** and receive messages via an HTTP endpoint.
3. **Monitor Pub/Sub message delivery** using **Cloud Logging**.

Review Questions:

1. What are the key components of Cloud Pub/Sub?
2. How does **push vs pull delivery** work?
3. What is the purpose of **dead letter topics (DLT)**?
4. How can **IAM roles** secure Pub/Sub access?
5. What are **real-world applications** of Cloud Pub/Sub?

Conclusion: Real-Time Event Processing with Cloud Pub/Sub

✔ **Cloud Pub/Sub enables scalable, real-time event-driven architecture**.
✔ **Supports push & pull subscriptions for flexible message consumption**.
✔ **Seamlessly integrates with Google Cloud services** for **big data processing & microservices**.

🚀 Mastering Cloud Pub/Sub helps build highly responsive, scalable cloud applications!

Cloud Composer – Managed Workflow Automation

Chapter 1: Introduction to Cloud Composer

1.1 What is Cloud Composer?

Cloud Composer is a **fully managed workflow orchestration service** built on **Apache Airflow**. It allows users to **create, schedule, and monitor data pipelines** in the cloud.

1.2 Key Features of Cloud Composer

✓ **Managed Apache Airflow** – No need to manage infrastructure.
✓ **Scalability** – Automatically scales based on workflow demand.
✓ **Hybrid & Multi-Cloud Support** – Connects to on-premises and external cloud services.
✓ **Built-in Security** – Uses **IAM and VPC Service Controls** for security.
✓ **Monitoring & Logging** – Integrated with **Cloud Logging and Cloud Monitoring**.

📌 **Example:**
A **financial services company** uses Cloud Composer to automate **data pipeline workflows** for fraud detection.

Chapter 2: Cloud Composer Architecture

2.1 Key Components

Component	Description

DAG (Directed Acyclic Graph)	Defines the sequence of tasks in a workflow.
Scheduler	Determines when and how workflows run.
Workers	Execute the tasks defined in DAGs.
Airflow Metadata Database	Stores workflow history and task execution logs.
Cloud Storage Bucket	Stores DAG files and workflow logs.

📌 **Example:**

A **retail company** uses DAGs in Cloud Composer to automate **daily sales report generation**.

Chapter 3: Setting Up Cloud Composer

3.1 Prerequisites

✓ **Google Cloud Project** with **billing enabled**.

✓ Enable the **Cloud Composer API**.

✓ Assign IAM roles: Composer Admin, Viewer, and Service Account User.

3.2 Deploying a Cloud Composer Environment

Step 1: Open Cloud Composer in Google Cloud Console

1. Navigate to **Cloud Composer** → **Environments**.
2. Click **Create Environment**.
3. Choose a **region** and specify **environment name**.

Step 2: Configure Composer Environment

✓ **Machine Type** – Choose **CPU & Memory allocation**.
✓ **Networking** – Select **VPC and Subnet**.
✓ **Python Packages** – Install additional **Python libraries (if needed)**.

📌 **Example:**
A **healthcare company** deploys Cloud Composer to **automate patient data synchronization** across hospitals.

Chapter 4: Creating Workflows in Cloud Composer

4.1 Understanding DAGs in Apache Airflow

A **DAG (Directed Acyclic Graph)** is a collection of tasks defining **workflow execution order**.

4.2 Creating a Simple DAG in Cloud Composer

1. **Upload DAG to Cloud Storage**
2. **Edit DAG in Python**
3. **Deploy DAG to Cloud Composer**

Example DAG: Hello World Workflow

```
from airflow import DAG

from airflow.operators.dummy_operator import DummyOperator

from datetime import datetime


default_args = {

  'owner': 'airflow',

  'start_date': datetime(2024, 1, 1),

  'retries': 1,
}


dag = DAG('hello_world',
          default_args=default_args,
          description='A simple Hello World DAG',
          schedule_interval='@daily')


start = DummyOperator(task_id='start', dag=dag)
end = DummyOperator(task_id='end', dag=dag)


start >> end  # Defines the sequence
```

📌 **Example:**

A **media company** automates **video processing workflows** using Cloud Composer DAGs.

CHAPTER 5: INTEGRATING CLOUD COMPOSER WITH OTHER SERVICES

5.1 Common Cloud Composer Integrations

Service	**Use Case**
BigQuery	Automate data transformation jobs.
Cloud Storage	Move files between cloud and on-premises.
Cloud Pub/Sub	Trigger workflows on real-time events.
Dataproc	Manage Spark and Hadoop jobs.

Vertex AI	Schedule ML model training workflows.

Example: Automating BigQuery Data Pipeline

```
from airflow.providers.google.cloud.operators.bigquery import BigQueryOperator


bq_query = BigQueryOperator(
   task_id='bq_query_task',
   sql='SELECT * FROM my_project.my_dataset.my_table',
   use_legacy_sql=False,
   dag=dag,
)
```

📌 **Example:**
A **banking institution** automates **customer transaction analysis** using Cloud Composer and BigQuery.

CHAPTER 6: MONITORING & DEBUGGING WORKFLOWS

6.1 Using Cloud Composer Logs & Monitoring

✓ **Airflow UI** – View DAG execution status.
✓ **Cloud Logging** – Track task failures and warnings.
✓ **Cloud Monitoring** – Set up alerts for workflow failures.

6.2 Debugging Failed Workflows

✓ Check **Airflow UI logs** for task failures.
✓ Use **retry mechanisms** in DAGs.
✓ Validate **permissions and IAM roles** for external services.

📌 **Example:**

A **social media company** monitors **ad performance workflows** in Cloud Composer.

Chapter 7: Best Practices for Cloud Composer

✓ **Use Environment Variables** – Store **API keys and credentials securely**.

✓ **Optimize DAG Execution** – Break large workflows into **modular tasks**.

✓ **Use Airflow Sensors** – Wait for external events before triggering tasks.

✓ **Leverage Airflow XComs** – Share data between tasks dynamically.

Example: Using XCom to Pass Data Between Tasks

```
from airflow.operators.python_operator import PythonOperator


def process_data(**kwargs):
    return 'processed_data'


task1 = PythonOperator(
    task_id='task1',
    python_callable=process_data,
    provide_context=True,
    dag=dag
)
```

📌 **Example:**

A **gaming company** automates **daily user engagement reports** using optimized DAG execution.

Chapter 8: Exercise & Review Questions

Exercise:

1. **Deploy a Cloud Composer Environment** in Google Cloud.
2. **Create a DAG** to automate a **file transfer workflow**.
3. **Integrate Cloud Composer with BigQuery** for data analysis.
4. **Set up logging & monitoring** to track DAG performance.

Review Questions:

1. What is the role of **DAGs in Cloud Composer**?
2. How does **Cloud Composer integrate with BigQuery**?
3. What are **best practices for debugging failed workflows**?
4. How does **IAM improve security in Cloud Composer**?
5. Why is **Airflow XCom used in DAGs**?

Conclusion: Automating Workflows with Cloud Composer

✔ **Cloud Composer automates complex workflows** using **Apache Airflow**.

✔ **Seamless integrations** with **BigQuery, Cloud Storage, and Dataproc**.

✔ **Monitoring tools like Cloud Logging and Monitoring** ensure **workflow reliability**.

🚀 Mastering Cloud Composer helps organizations build scalable, automated data pipelines efficiently!

GOOGLE DATA STUDIO – DATA VISUALIZATION & REPORTING

CHAPTER 1: INTRODUCTION TO GOOGLE DATA STUDIO

1.1 What is Google Data Studio?

Google Data Studio is a **free, cloud-based data visualization tool** that helps users create **interactive reports and dashboards**. It enables businesses to **connect, visualize, and share data from multiple sources** with an easy-to-use interface.

1.2 Key Features of Google Data Studio

✓ **Drag-and-Drop Interface** – No coding required.

✓ **Connects to Multiple Data Sources** – Supports Google Analytics, BigQuery, Sheets, SQL databases, and more.

✓ **Customizable Reports & Dashboards** – Offers filters, charts, and interactive elements.

✓ **Real-Time Data Updates** – Ensures up-to-date reporting.

✓ **Collaboration & Sharing** – Similar to Google Docs, allowing multiple users to edit reports.

📌 **Example:**

A **marketing team** uses **Google Data Studio to create real-time dashboards** for tracking **website traffic and conversion rates** from Google Analytics.

CHAPTER 2: CONNECTING DATA SOURCES IN GOOGLE DATA STUDIO

2.1 Supported Data Sources

Google Data Studio supports both **Google and third-party data sources**.

Google Data Sources	**Third-Party Data Sources**
Google Analytics	MySQL, PostgreSQL, SQL Server
Google Ads	Facebook Ads
Google BigQuery	Salesforce
Google Sheets	Shopify
Google Search Console	Stripe, Mailchimp

2.2 How to Connect a Data Source

1. Open **Google Data Studio** (https://datastudio.google.com).
2. Click **Create** → **Data Source**.
3. Select a **data connector** (e.g., Google Sheets, BigQuery).
4. Grant **permissions** and authorize access.
5. Configure **data fields and metrics**.
6. Click **Add to Report**.

📌 **Example:**

A **finance team** connects **Google Sheets to Data Studio** to generate **monthly financial reports** dynamically.

Chapter 3: Creating Data Visualizations

3.1 Common Chart Types in Data Studio

Chart Type	**Usage**

Bar Chart	Compare categorical data (e.g., sales by region).
Line Chart	Show trends over time (e.g., website traffic).
Pie Chart	Display proportions (e.g., revenue by product category).
Geo Map	Visualize location-based data (e.g., user traffic by country).
Tables	Present structured data (e.g., product sales).

3.2 Steps to Create a Chart in Google Data Studio

1. Open a **report** in Google Data Studio.
2. Click **+ Add a chart**.
3. Select a **chart type** (e.g., Bar, Line, Pie).
4. Choose the **data source**.
5. Configure **dimensions (categories)** and **metrics (values)**.
6. Customize the **colors, labels, and styles**.
7. Click **Apply** to add it to the dashboard.

📌 **Example:**
A **sales team** creates a **line chart in Data Studio** to track **monthly revenue growth trends**.

CHAPTER 4: CUSTOMIZING REPORTS & DASHBOARDS

4.1 Adding Filters & Controls

Filters allow users to **interact with reports dynamically**.

✓ **Date Filters** – Enable users to view data for different time periods.
✓ **Dropdown Filters** – Allow filtering by product, region, or

department.

✓ **Search Filters** – Let users search specific data points.

4.2 Steps to Add Filters

1. Click **+ Add a Control** in the report.
2. Choose **Date Range, Drop-down List, or Search Box**.
3. Select the **data source** and assign relevant fields.
4. Adjust styling and apply changes.

📌 **Example:**

A **retail company** adds **filters to a sales dashboard** to let users **select different product categories**.

4.3 Using Calculated Fields for Custom Metrics

Calculated fields allow users to create **custom formulas** for advanced analysis.

✓ **Example Formulas:**

- **Profit Margin:** (Revenue - Cost) / Revenue
- **Click-Through Rate (CTR):** (Clicks / Impressions) * 100
- **Customer Retention Rate:** (Returning Customers / Total Customers) * 100

Steps to Create a Calculated Field

1. Go to the **Data Source** settings.
2. Click **+ Add a Field**.
3. Enter a **Formula (e.g., Revenue - Expenses)**.
4. Click **Apply** and use it in visualizations.

📌 **Example:**
A **digital marketing agency** creates a **CTR metric** using calculated fields to track **advertising campaign performance**.

Chapter 5: Sharing & Collaborating on Reports

5.1 How to Share a Google Data Studio Report

1. Click **Share** in the top-right corner.
2. Add **email addresses** of collaborators.
3. Set permissions:
   - **Viewer (Read-only)**
   - **Editor (Can edit and modify reports)**
4. Click **Send Invitation**.

📌 **Example:**
A **CEO shares a performance dashboard** with department heads to track **monthly KPIs**.

5.2 Embedding & Exporting Reports

✔ **Embed in Websites** – Copy **iframe code** to embed reports in a website.
✔ **Export as PDF** – Download reports for offline analysis.
✔ **Schedule Email Reports** – Send **automated reports** to stakeholders.

📌 **Example:**
A **news website** embeds **Google Data Studio charts** to show **live election results**.

Chapter 6: Best Practices for Data Visualization in Data Studio

✓ **Keep Reports Simple & Clear** – Avoid overloading dashboards with too much data.

✓ **Use Consistent Colors & Formatting** – Maintain uniform styling for better readability.

✓ **Highlight Key Metrics with Scorecards** – Show important KPIs at the top.

✓ **Use Filters to Enhance Interactivity** – Allow users to explore data dynamically.

✓ **Optimize Performance** – Reduce load time by using **pre-aggregated data sources**.

📌 **Example:**

A **marketing manager** designs a **minimalistic dashboard** with **key performance indicators (KPIs) at the top**.

Chapter 7: Exercise & Review Questions

Exercise:

1. **Create a sales report in Google Data Studio** using Google Sheets.
2. **Add a pie chart** showing **sales distribution by region**.
3. **Implement filters** for date range and product categories.
4. **Share the report with your team** and schedule automated email updates.

Review Questions:

1. What is the main purpose of Google Data Studio?

2. How can you connect **Google Analytics** to Google Data Studio?
3. What are **calculated fields**, and how can they be used?
4. What is the difference between **a bar chart and a pie chart**?
5. How do you **share a report with collaborators** in Google Data Studio?

Conclusion: Mastering Data Visualization with Google Data Studio

✓ **Google Data Studio simplifies business intelligence** with **real-time, interactive dashboards**.
✓ **Connect multiple data sources**, create **custom metrics**, and build **engaging visual reports**.
✓ **Collaboration and automation** make reporting efficient for **business teams and analysts**.

🚀 **Mastering Google Data Studio enables professionals to transform raw data into actionable insights!**

ASSIGNMENT

PROCESS STREAMING DATA USING CLOUD DATAFLOW

SOLUTION: PROCESS STREAMING DATA USING CLOUD DATAFLOW

Overview

Google **Cloud Dataflow** is a **fully managed stream and batch data processing service** built on **Apache Beam**. It enables **real-time analytics, ETL (Extract, Transform, Load) workflows, and AI/ML model streaming** by processing live data from sources like **Pub/Sub, Kafka, and Cloud Storage**.

Step 1: Set Up the Google Cloud Environment

1.1 Enable Required Services

Ensure the following services are **enabled** in Google Cloud:

```
gcloud services enable dataflow.googleapis.com pubsub.googleapis.com storage.googleapis.com bigquery.googleapis.com
```

1.2 Set Up IAM Permissions

Assign the **necessary roles** to the user or service account:

- **Dataflow Admin**
- **Pub/Sub Editor**
- **BigQuery Data Editor**
- **Cloud Storage Admin**

```
gcloud projects add-iam-policy-binding [PROJECT_ID] \
  --member=user:[YOUR_EMAIL] --role=roles/dataflow.admin
```

📌 **Example:**

A **log monitoring system** assigns IAM roles to **Dataflow processing agents** to handle real-time logs.

Step 2: Create a Streaming Data Source (Pub/Sub)

2.1 Create a Pub/Sub Topic & Subscription

Create a **Pub/Sub topic** to receive streaming data:

```
gcloud pubsub topics create streaming-topic
```

Create a **subscription** to allow Dataflow to pull messages:

```
gcloud pubsub subscriptions create streaming-sub \
  --topic=streaming-topic
```

📌 **Example:**

A **retail company** sets up a **Pub/Sub topic** to receive **real-time sales transactions**.

Step 3: Develop a Streaming Pipeline Using Apache Beam

3.1 Install Apache Beam SDK

Ensure you have **Apache Beam** installed:

```
pip install apache-beam[gcp]
```

3.2 Write a Python Pipeline to Process Streaming Data

Create a script **dataflow_streaming.py**:

```
import apache_beam as beam

from apache_beam.options.pipeline_options import PipelineOptions, StandardOptions
```

```
# Define pipeline options
pipeline_options = PipelineOptions(
    streaming=True,
    project='[PROJECT_ID]',
    region='us-central1',
    job_name='streaming-dataflow-job',
    temp_location='gs://[BUCKET_NAME]/temp',
    staging_location='gs://[BUCKET_NAME]/staging'
)

# Define a processing function
def transform_data(element):
    import json
    record = json.loads(element)
    return {'user_id': record['user_id'], 'purchase_amount': record['amount']}

# Build the Dataflow pipeline
with beam.Pipeline(options=pipeline_options) as pipeline:
    (
        pipeline
        | 'Read from Pub/Sub' >> beam.io.ReadFromPubSub(topic='projects/[PROJECT_ID]/topics/streaming-topic')
        | 'Decode UTF-8' >> beam.Map(lambda msg: msg.decode('utf-8'))
        | 'Transform Data' >> beam.Map(transform_data)
        | 'Write to BigQuery' >> beam.io.WriteToBigQuery(
          '[PROJECT_ID]:dataset_name.table_name',
          schema='user_id:STRING, purchase_amount:FLOAT',
write_disposition=beam.io.BigQueryDisposition.WRITE_APPEND
        )
    )
```

📌 **Example:**
A **banking application** processes **real-time transactions** and writes the output to **BigQuery for fraud detection**.

Step 4: Deploy the Streaming Pipeline to Cloud Dataflow

4.1 Upload Python Script to Cloud Storage

```
gsutil cp dataflow_streaming.py gs://[BUCKET_NAME]/dataflow/
```

4.2 Run Dataflow Job Using Cloud CLI

```
python -m apache_beam.runners.dataflow.dataflow_runner \
  --project=[PROJECT_ID] \
  --region=us-central1 \
  --staging_location=gs://[BUCKET_NAME]/staging/ \
  --temp_location=gs://[BUCKET_NAME]/temp/ \
  --runner=DataflowRunner
```

📌 **Example:**
A **stock trading company** deploys a Dataflow job to **track real-time stock price changes**.

Step 5: Monitor and Optimize Streaming Jobs

5.1 Monitor Dataflow Pipeline

- Open **Google Cloud Console → Dataflow**
- View **job status**, logs, and data throughput
- Use **Cloud Logging** to troubleshoot failures

```
gcloud dataflow jobs list
```

5.2 Optimize Streaming Jobs

✓ **Autoscaling**: Use **flexible worker count** to optimize resources
✓ **Use Dataflow Shuffle**: Reduces memory footprint for large datasets
✓ **Dead-letter queue**: Handle **failed messages** by routing them to **Pub/Sub or Cloud Storage**

📌 **Example:**
A **smart home automation** company optimizes streaming jobs **to scale up processing during peak hours**.

Step 6: Exercise & Review Questions

Exercise:

1. **Create a Pub/Sub topic** and publish test messages.

2. **Deploy a streaming Dataflow pipeline** that writes to BigQuery.
3. **Optimize Dataflow performance** using autoscaling.

Review Questions:

1. What are the **benefits of Cloud Dataflow** for streaming data?
2. How does **Pub/Sub work as a data ingestion source** for Dataflow?
3. What are the **best practices for scaling Dataflow jobs**?
4. How can **Cloud Monitoring help optimize streaming pipelines**?
5. How does **BigQuery integration enhance real-time analytics**?

CONCLUSION: CLOUD DATAFLOW FOR REAL-TIME PROCESSING

✓ Fully managed Apache Beam execution for streaming & batch workloads

✓ Integrates with Pub/Sub, BigQuery, Cloud Storage, and AI services

✓ Autoscaling & fault-tolerant streaming for enterprise-grade applications

✓ Secure, cost-effective, and highly scalable solution for big data analytics

🚀 **Mastering Cloud Dataflow enables real-time insights and data-driven decision-making!**

ANALYZE LARGE DATASETS USING BIGQUERY

SOLUTION: ANALYZE LARGE DATASETS USING BIGQUERY

Step 1: Understanding BigQuery

1.1 What is BigQuery?

Google BigQuery is a **serverless, highly scalable, and cost-effective data warehouse** designed for **fast SQL-based analytics** on large datasets.

✓ **Serverless** – No infrastructure management required.
✓ **Massive Scalability** – Handles **petabyte-scale** data analysis.
✓ **Real-time Analytics** – Supports **streaming data ingestion**.
✓ **Built-in AI & ML Capabilities** – Runs **machine learning models** directly in SQL.

📌 **Example:**
A **retail company** uses BigQuery to **analyze customer purchase trends** from millions of transactions.

Step 2: Setting Up BigQuery

2.1 Prerequisites

✓ A **Google Cloud account** with billing enabled.
✓ Enable **BigQuery API** in the Google Cloud Console:

```
gcloud services enable bigquery.googleapis.com
```

✓ Install and authenticate the **Google Cloud SDK**:

```
gcloud auth login
```

2.2 Creating a BigQuery Dataset

1. Open **Google Cloud Console** → Navigate to **BigQuery**.
2. Click **Create Dataset** → Enter sales_dataset.
3. Choose a **location** (e.g., US).
4. Click **Create**.

📌 **Example:**

A **finance company** creates a dataset named transactions_dataset to store stock market transaction records.

Step 3: Importing Data into BigQuery

3.1 Loading Data from Google Cloud Storage

1. Go to **BigQuery** → Select your **dataset (sales_dataset)**.
2. Click **Create Table** → Choose **Source: Cloud Storage**.
3. Enter the Cloud Storage URI:
4. gs://my-bucket/sales_data.csv
5. Select **File Format: CSV**.
6. Click **Create Table**.

3.2 Loading Data from a Local File

Using **gcloud CLI**, upload a CSV file:

bq load --source_format=CSV sales_dataset.sales_table ./sales_data.csv

3.3 Loading Data from Google Sheets

1. In **BigQuery Console**, click **Create Table**.
2. Select **Source: Google Drive** → Enter the Google Sheet URL.

3. Choose **File Format: Google Sheets**.
4. Click **Create Table**.

📌 **Example:**
An **IoT company** loads **sensor data** into BigQuery for **real-time temperature monitoring**.

Step 4: Querying Large Datasets in BigQuery

4.1 Running a Simple Query

Example: Get total sales per region

```
SELECT region, SUM(sales) AS total_sales
FROM `sales_dataset.sales_table`
GROUP BY region
ORDER BY total_sales DESC;
```

4.2 Filtering Data Efficiently

Example: Find orders over $500 in the last month

```
SELECT order_id, customer_name, total_amount
FROM `sales_dataset.sales_table`
WHERE total_amount > 500
AND order_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 30 DAY);
```

4.3 Using Partitioning for Faster Queries

```
CREATE TABLE sales_dataset.sales_partitioned
PARTITION BY DATE(order_date) AS
```

```
SELECT * FROM `sales_dataset.sales_table`;
```

📌 **Example:**
A **healthcare company** runs queries on **partitioned patient records** to retrieve **last month's appointments**.

Step 5: Optimizing Queries for Performance

5.1 Best Practices for Query Optimization

✓ **Use Partitioned Tables** – Speeds up queries on large datasets.
✓ **Avoid SELECT *** – Query only the required columns.
✓ **Use Filter Conditions** – Reduces scanned data.
✓ **Leverage Clustering** – Groups data to improve indexing.

📌 **Example:**
A **media company** uses **partitioned logs** to **quickly analyze user behavior** on their streaming platform.

Step 6: Visualizing Data in Google Data Studio

6.1 Connecting BigQuery to Data Studio

1. Open **Google Data Studio** → Click **Create Report**.
2. Click **Add Data** → Select **BigQuery**.
3. Choose sales_dataset.sales_table.
4. Click **Add to Report**.
5. Create **charts & dashboards** to visualize **sales trends**.

📌 **Example:**
A **marketing team** visualizes **advertising performance** using a **BigQuery-powered dashboard**.

Step 7: Automating Data Pipelines with BigQuery

7.1 Scheduling Queries in BigQuery

1. In **BigQuery Console**, go to **Scheduled Queries**.
2. Click **Create a New Scheduled Query**.
3. Enter SQL Query:
4. SELECT customer_id, COUNT(order_id) AS order_count
5. FROM `sales_dataset.sales_table`
6. WHERE order_date >= CURRENT_DATE()
7. GROUP BY customer_id;
8. Set **Run Frequency** (e.g., **Daily at Midnight**).
9. Click **Create**.

📌 **Example:**
An **HR team** sets up **automated reporting** for **employee attendance data**.

Step 8: Using BigQuery ML for Predictive Analysis

8.1 Training a Machine Learning Model in BigQuery

Predict customer churn using a logistic regression model:

```
CREATE OR REPLACE MODEL sales_dataset.churn_model

OPTIONS(model_type='logistic_reg') AS

SELECT age, total_spent, visit_frequency, churn_label

FROM `sales_dataset.sales_table`;
```

8.2 Running Predictions

```
SELECT customer_id, predicted_churn

FROM ML.PREDICT(MODEL sales_dataset.churn_model,

 (SELECT age, total_spent, visit_frequency FROM `sales_dataset.sales_table`));
```

📌 **Example:**
A **telecom company** builds a **churn prediction model** in BigQuery ML to identify customers likely to switch providers.

Step 9: Exercise & Review Questions

Exercise:

1. **Create a BigQuery dataset** and import a sample CSV file.
2. **Run SQL queries** to analyze sales performance.
3. **Set up a scheduled query** to automate reporting.
4. **Create a dashboard in Google Data Studio** using BigQuery data.
5. **Train a machine learning model** using BigQuery ML.

Review Questions:

1. What are the advantages of **serverless BigQuery** over traditional databases?
2. How does **partitioning and clustering** improve query performance?
3. How can **BigQuery ML** be used for predictive analytics?
4. What are the different ways to **import data into BigQuery**?

5. How does **Google Data Studio** enhance BigQuery data visualization?

CONCLUSION: SCALING ANALYTICS WITH BIGQUERY

✓ BigQuery enables fast and scalable data analysis.
✓ Supports structured, semi-structured, and streaming data.
✓ Built-in ML capabilities allow predictive analytics with SQL.
✓ Integrates seamlessly with Google Data Studio for visualization.

🚀 **Mastering BigQuery helps businesses unlock insights from large datasets!**