03 Data Engineering - GCP X AWS -Nikhil Sharma

1: Fundamentals of SQL: Basic to Advanced

1.1 Introduction to SQL

- What is SQL? Importance in Data Engineering
- Relational databases: Tables, rows, columns, keys (primary, foreign, surrogate, etc.)
- ERD, Referential Integrity and Normalization
- Basic SQL syntax: `SELECT`, `FROM`, `WHERE`, `ORDER BY`
- Sub Languages: `DDL`, `DML`, `DQL`, `DCL`, `TCL`

1.2 Data Retrieval

- Filtering data: WHERE clause, comparison operators, logical operators
- Sorting data: `ORDER BY`, `ASC/DESC`
- Aggregations: `COUNT`, `SUM`, `AVG`, `MIN`, `MAX`
- Grouping data: `GROUP BY`, `HAVING`

1.3 Joins and Relationships

- `INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN`, `FULL OUTER JOIN`
- Self-joins and cross joins
- Handling NULL values

1.4 Data Cleaning and Transformation

- Handling missing data: `COALESCE`, `CASE` statements
- String manipulation: `CONCAT`, `SUBSTRING`, `REPLACE`
- Date and time functions: `DATE_FORMAT`, `TIMESTAMPDIFF`
- Various Scalar Functions

1.5 Advanced SQL

- Subqueries and nested queries
- Common Table Expressions (CTEs)
- Temporary tables and views
- Window functions: `ROW_NUMBER`, `RANK`, `DENSE_RANK`, `NTILE`, `LEAD`, `LAG`, etc.

1.6 Performance Optimization

- Query execution plans
- Indexing strategies: B-trees
- Indexing and query optimization

1.7 Integrating SQL with Python

- Using libraries like `sqlite3`, `SQLAlchemy`, etc.
- Automating SQL queries and data pipelines

Project: SQL Retail Database Analysis Project (Northwind)

2: Fundamentals of Python: Basic to Advanced

2.1 Introduction to Python

- Why Python? Its role in Data Engineering and Data Science
- Setting up the environment: Anaconda, Jupyter Notebook, IDEs
- Basic syntax: Variables, data types, operators, formatting

2.2 Control Structures

- Conditional statements: `if`, `elif`, `else`
- Loops: for, while, break, continue, pass

2.3 Data Structures

- Lists, tuples, sets, dictionaries
- Mutable vs. immutable objects
- Nested data structures
- List comprehensions, dictionary comprehensions, etc.

2.4 Functions and Modules

- Defining and calling functions
- Scope and namespaces
- Types of arguments, `*args`, `**kwargs`
- Importing modules and libraries
- Creating packages and modules

2.5 Exception Handling

- Errors Vs Exceptions
- `try` `except` `finally` blocks
- Custom exceptions

2.6 Object-Oriented Programming (OOP)

- Classes and objects, `__init__`, `__str__`, `__repr__`, `__eq__`, etc.
- Inheritance, polymorphism, encapsulation, abstraction
- Magic methods and operator overloading

2.8 File Handling

- File operations, `os`, `shutil`
- File-paths: `os`, `pathlib`, `glob`
- Reading and writing files: CSV, JSON
- Working with APIs: Making HTTP requests, parsing responses

2.9 Libraries for Data Manipulation and Viz

- NumPy: Arrays, matrix operations, broadcasting
- Pandas: DataFrames, data manipulation, aggregation
- Matplotlib/Seaborn: Data visualization

Project: Retail Sales Analysis Project - Superstore Sales Dataset

3. Introduction to Big Data and Hadoop Fundamentals

3.1 Big Data Fundamentals:

- What is Big Data? Characteristics (Volume, Velocity, Variety, Veracity, Value)
- Components of Big Data: Generation, Collection, Processing, Storage, Analysis
- Benefits and Challenges of Big Data

3.2 Hadoop Introduction:

- Evolution of Hadoop and its role in Big Data
- Hadoop Architecture: HDFS, MapReduce, YARN
- HDFS Architecture: NameNode, DataNode, Secondary NameNode
- Basic Hadoop Ecosystem Components: HDFS, MapReduce, YARN, Common Utilities
- Hadoop Data Formats: Text Files, Sequence Files, Avro, Parquet, ORC

1. **Project 1: Setting Up a Hadoop Cluster**
   - **Objective:** Install and configure a single-node Hadoop cluster on your local machine or cloud.
   - **Skills Learned:** Understanding Hadoop architecture, setting up HDFS, and basic file operations.
2. **Project 2: Exploring HDFS Commands**
   - **Objective:** Perform basic HDFS operations (e.g., creating directories, uploading/downloading files) using the command line.
   - **Skills Learned:** Navigating HDFS, managing data storage, and understanding file formats.

4: Introduction to Cloud Computing with AWS & GCP

4.1 Cloud Computing Fundamentals:

- Cloud models: IaaS, PaaS, SaaS
- Deployment models: Public, Private, Hybrid
- Benefits and risks of cloud computing

4.2 AWS Global Infrastructure:

- Regions, Availability Zones (AZs), Edge Locations
- Service availability and reliability

4.3 GCP Global Infrastructure:

- Regions, Zones, Edge Network
- Service availability and reliability

4.4 AWS Basic Services:

- Compute: EC2 basics, AMI, Auto Scaling

- Storage: S3 basics, EBS
- IAM Basics: Users, Groups, Roles, Policies

4.5 GCP Basic Services:

- Compute: Compute Engine, Machine types, Instance templates
- Storage: Cloud Storage, Persistent Disk
- IAM Basics: Service accounts, Roles, Permissions

Projects:

1. **Project 1A: Setting Up an AWS Account and Launching an EC2 Instance**
   - **Objective:** Create an AWS account, launch an EC2 instance, and connect to it via SSH.
   - **Skills Learned:** Understanding AWS services, launching and managing compute resources.
2. **Project 1B: Setting Up a GCP Account and Launching a Compute Engine Instance**
   - **Objective:** Create a GCP account, launch a Compute Engine instance, and connect to it via SSH.
   - **Skills Learned:** Understanding GCP services, launching and managing compute resources.
3. **Project 2A: Storing and Retrieving Data from AWS S3**
   - **Objective:** Upload files to an S3 bucket, retrieve them programmatically, and manage access permissions.
   - **Skills Learned:** Using S3 for storage, managing permissions, and interacting with AWS CLI.
4. **Project 2B: Storing and Retrieving Data from GCP Cloud Storage**
   - **Objective:** Upload files to a Cloud Storage bucket, retrieve them programmatically, and manage access permissions.
   - **Skills Learned:** Using Cloud Storage, managing permissions, and interacting with gcloud CLI.

5: Spark Fundamentals and PySpark Basics

5.1 Spark Introduction:

- What is Spark? Spark vs. Hadoop
- Spark Ecosystem: Spark Core, Spark SQL, MLlib, GraphX, Streaming
- Spark Setup: Local mode vs. Cluster mode

5.2 PySpark Basics:

- Introduction to RDDs: Transformations and Actions
- SparkSession: Entry point for PySpark applications
- Reading and writing data in various formats (CSV, JSON, Parquet)
- Shared Variables: Broadcast variables and accumulators

5.3 Simple Project Exercises:

1. **Project 1: Word Count with PySpark**
   - **Objective:** Implement a word count program using PySpark to process a text file.
   - **Skills Learned:** Understanding RDDs, transformations, and actions in PySpark.
2. **Project 2: Analyzing a Dataset with PySpark**
   - **Objective:** Load a dataset (e.g., Titanic or Iris) into PySpark, perform basic transformations, and save the results.
   - **Skills Learned:** Working with PySpark DataFrames, performing data transformations, and saving outputs.

5.4 Spark SQL and DataFrames

5.5 Spark SQL DataFrames:

- Schema management and DataFrame operations
- Narrow and Wide Transformations
- Selecting, Renaming, Adding, Dropping columns
- Basic queries: Filtering, sorting, aggregations
- Window functions and UDFs (User-Defined Functions)

5.6 Optimization Techniques:

- Caching and persistence strategies
- Spark UI: Understanding DAGs, Jobs and Stages
- Partitioning strategies for better performance

5.7 Cloud Integration for Spark:

AWS INTEGRATION:

- Running Spark jobs on AWS EMR
- Integrating Spark with S3 for data storage
- Configuring IAM roles for Spark clusters
- AWS Glue for Spark ETL jobs

GCP INTEGRATION:

- Running Spark jobs on Dataproc
- Integrating Spark with Google Cloud Storage
- Configuring service accounts for Spark clusters
- Cloud Composer (managed Airflow) for orchestrating Spark jobs

6.4 Advanced Project Exercises:

1. **Project 1: Building a Data Pipeline with Spark SQL**
   - **Objective:** Use Spark SQL to analyze a dataset, perform aggregations, and generate insights.
   - **Skills Learned:** Writing Spark SQL queries, working with DataFrames, and optimizing performance.
2. **Project 2A: Deploying a Spark Job on AWS EMR**
   - **Objective:** Set up an EMR cluster, run a PySpark job, and store results in S3.
   - **Skills Learned:** Deploying Spark applications on AWS, integrating with AWS cloud storage.
3. **Project 2B: Deploying a Spark Job on GCP Dataproc**
   - **Objective:** Set up a Dataproc cluster, run a PySpark job, and store results in Cloud Storage.
   - **Skills Learned:** Deploying Spark applications on GCP, integrating with GCP cloud storage.
4. **Project 3: Multi-Cloud Spark Data Pipeline (Advanced)**
   - **Objective:** Create a data pipeline that processes data from one cloud provider and outputs results to another.
   - **Skills Learned:** Cross-cloud integration, understanding differences in configuration and optimization between cloud platforms.

CLOUD SERVICES FOR DATA ENGINEERS

6: Data Engineering on AWS

6.1 AWS Storage Foundations

- Amazon S3 as the data lake foundation
- S3 storage classes and lifecycle management
- S3 access patterns and performance optimization
- S3 security best practices
- S3 Select and Glacier retrieval options

6.2 AWS Glue

- Glue ETL in depth understanding and implementation
- Workflows and job bookmarks
- Execution types and resource allocation
- Data Quality with AWS Glue DataQuality
- Glue Databrew for visual data preparation
- Glue Schema Registry for schema evolution

6.3 Data Processing Options

- Amazon EMR architecture and components
- EMR storage options (HDFS, EMRFS)
- Creating and managing EMR clusters
- EMR Serverless for job-based workloads
- AWS Lambda for lightweight data transformations
- Cost optimization strategies for data processing

6.4 Data Warehousing with Amazon Redshift

- Architecture and node types
- Distribution styles and sort keys for performance
- Query tuning and workload management
- Redshift Spectrum for data lake querying
- Redshift Serverless implementation
- Materialized views and result caching
- Data sharing and cross-database queries

6.5 Building and Managing Data Lakes

- Data lake vs data warehouse architectures
- AWS Lake Formation for data lake management
- Data cataloging with Glue Data Catalog
- Managing permissions and access control
- Open table formats (Parquet, ORC, Delta, Iceberg, etc.)
- Data governance and quality enforcement

6.6 Streaming Data Processing

- Amazon Kinesis Data Streams fundamentals
- Kinesis Data Firehose for delivery
- Kinesis Data Analytics for real-time processing
- Integration patterns with other AWS services
- Exactly-once processing strategies

6.7 Data Analysis and Visualization

- Amazon Athena for SQL queries against S3
- Amazon QuickSight visualization capabilities
- QuickSight integration with AWS data sources
- Creating interactive dashboards
- Embedding analytics in applications

6.8 Orchestrating Data Pipelines

- AWS Step Functions for workflow management
- Amazon MWAA (Managed Airflow)
- EventBridge for event-driven pipelines
- Pipeline monitoring and error handling
- Amazon AppFlow for SaaS integration
- AWS Data Exchange for data products

6.9 Security and Governance

- Identity and access management for data services
- Encryption options across the data pipeline
- AWS CloudTrail for audit logging
- Sensitive data detection with Macie
- Implementing data governance frameworks

6.10 Real-World Architectures

- Batch processing architectures
- Real-time analytics architectures
- Cost optimization and Performance tuning strategies

---

7 Data Engineering on Google Cloud Platform (GCP)

1: GCP Storage Foundations

1.1 Cloud Storage

- Storage classes and lifecycle management
- Access control and permissions
- Performance optimization patterns
- Data transfer services and tools
- Storage insights and analytics

1.2 Cloud SQL and Cloud Spanner

- Relational database options on GCP
- Cloud SQL deployment and management
- Spanner architecture and global distribution
- High availability and disaster recovery
- Migration strategies from on-premises databases

2: Data Processing and ETL

2.1 Cloud Dataflow

- Apache Beam programming model
- Batch vs. streaming pipelines
- Template creation and management
- Custom Python pipelines
- Performance optimization and monitoring
- Dataflow SQL for SQL-based transformations

2.2 Cloud Dataproc

- Managed Hadoop and Spark environment
- Cluster creation and scaling
- Job submission and management
- Integration with Cloud Storage
- Dataproc Serverless for job-based workloads
- Cost optimization strategies

2.3 Cloud Data Fusion

- No-code/low-code data integration
- Pipeline development and debugging
- Creating reusable plugins and templates
- Metadata management
- Pipeline monitoring and lineage

3: Data Warehousing

3.1 BigQuery Fundamentals

- Architecture and storage model
- Dataset and table management
- Loading and exporting data
- Query optimization and performance tuning
- Slots and reservation model

3.2 Advanced BigQuery Features

- BigQuery ML for in-database machine learning
- Materialized views and query optimization
- Partitioning and clustering strategies
- Data governance and column-level security
- Data sharing and Analytics Hub

3.3 Connected Sheets and BI Tools

- Integration with Google Sheets
- Looker Studio (formerly Data Studio) dashboards
- Looker for enterprise BI
- Third-party tool integration

4: Data Lakes on GCP

4.1 Building Modern Data Lakes

- Data lake architecture patterns
- Cloud Storage as data lake foundation
- Organizing data for performance and governance
- Metadata management with Dataplex
- Open table formats (Parquet, ORC, Avro)

4.2 Dataplex

- Data mesh implementation with Dataplex
- Creating lakes, zones, and assets
- Discovering and exploring data
- Quality monitoring and metrics
- Access control and governance

5: Streaming Data Processing

5.1 Pub/Sub

- Messaging architecture and guarantees
- Topic and subscription management
- Push vs. pull delivery models
- Message filtering and ordering
- Exactly-once processing strategies

5.2 Dataflow for Streaming

- Streaming pipeline patterns
- Windowing and watermark concepts
- State management and fault tolerance
- Handling late data
- Streaming analytics

6: Orchestration and Workflow Management

6.1 Cloud Composer

- Managed Apache Airflow environment
- DAG development and deployment
- Scheduling and triggering workflows
- Monitoring and troubleshooting

- Managing environment variables and connections

6.2 Workflows

- Serverless workflow orchestration
- Integration with GCP and external services
- Error handling and retries
- State management
- CI/CD for workflow deployment

7: Data Governance and Security

7.1 Data Catalog

- Metadata management and discovery
- Technical and business metadata
- Schema management and evolution
- Tagging and classification

7.2 Security and Access Control

- IAM for data services
- Data access patterns and best practices
- VPC Service Controls for network isolation
- Encryption options (CMEK, default encryption)
- Sensitive data protection with DLP

7.3 Multi-cloud

- BigQuery Omni for multi-cloud analytics

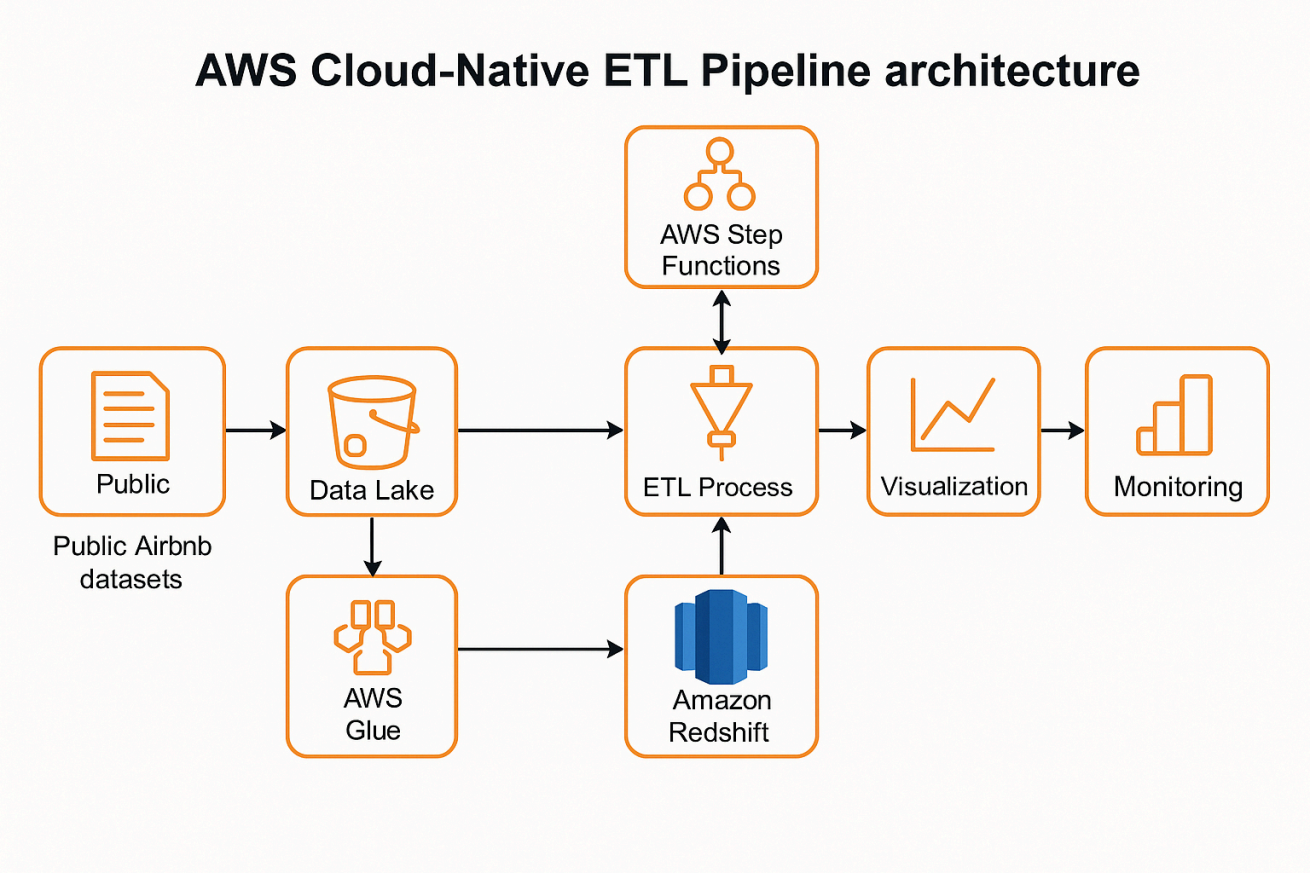
Capstone Projects

1. AWS Cloud-Native ETL Pipeline for Hospitality Analytics

Project Overview

This capstone project implements a fully automated cloud-native ETL pipeline for analyzing hospitality rental data (Airbnb) using AWS services. The solution enables data-driven decision making with near real-time insights into market trends, pricing optimization, and property performance.

Architecture



Core Components

- **Data Source**: Public Airbnb datasets (listings, reviews, calendar) in CSV/JSON format
- **Data Lake**: Amazon S3 with partitioned storage strategy
- **ETL Processing**: AWS Glue with Spark jobs
- **Data Warehouse**: Amazon Redshift
- **Visualization**: Amazon QuickSight
- **Orchestration**: AWS Step Functions
- **Monitoring**: CloudWatch

Implementation Details

1. Data Ingestion Layer

- Configure S3 buckets with logical structure:

```
s3://hospitality-data-lake/
    ├── raw/
    │   ├── listings/{year}/{month}/{day}/
    │   ├── reviews/{year}/{month}/{day}/
    │   └── calendar/{year}/{month}/{day}/
    └── processed/
        ├── listings/{year}/{month}/{day}/
        ├── reviews/{year}/{month}/{day}/
        └── calendar/{year}/{month}/{day}/
```

2. Data Processing Layer

- Create Glue Data Catalog to maintain schema information
- Develop Glue ETL jobs for data transformation:
  - **Data Cleansing**: Handle missing values, standardize formats, remove duplicates
  - **Data Enrichment**: Calculate metrics like occupancy rate, pricing efficiency, seasonal indices
  - **Data Standardization**: Normalize location data, categorize property types
- Apply column-level encryption for sensitive data

3. Data Warehouse Layer

- Design optimized Redshift schema with appropriate distribution and sort keys
- Implement efficient loading strategy using COPY command with automatic compression
- Create materialized views for common analytical queries

4. Visualization Layer

- Develop QuickSight dashboards with key insights:
  - **Market Analysis**: Price trends by location, seasonality patterns
  - **Performance Metrics**: Occupancy rates, average daily rates
  - **Competitive Analysis**: Comparison against similar properties

5. Quality & Monitoring

- Implement data quality validation using AWS Deequ
- Set up CloudWatch alarms for pipeline health monitoring
- Create automated reconciliation checks between source and target

6. Automation & Orchestration

- Design Step Functions workflow to coordinate the entire pipeline
- Implement error handling with retry logic and failure notifications
- Schedule incremental updates with configurable frequency

Expected Outcomes

- Production-ready data pipeline with 99.9% reliability
- Analytical dashboards providing actionable business intelligence
- Documentation for maintenance and future enhancements
- Cost optimization through appropriate resource sizing

Technical Skills Demonstrated

- AWS cloud architecture design
- Data lake implementation
- ETL processing with Spark
- Data warehouse optimization
- Pipeline orchestration
- Data quality management
- Business intelligence visualization

Extension Possibilities

- Implement real-time analytics using Kinesis

- Add machine learning models for price prediction
- Develop APIs for external application integration

---

2. GCP E-Commerce Analytics Pipeline

Project Overview

This capstone project creates a scalable data integration and analytics platform for e-commerce shopping cart data using Google Cloud Platform services. The solution enables comprehensive analysis of customer behavior, product performance, and sales patterns to drive business growth.

Architecture

Core Components

- **Data Source**: MySQL database with e-commerce transaction data
- **Data Processing**: Dataflow and Dataproc
- **Data Storage**: Cloud Storage and BigQuery
- **Orchestration**: Cloud Composer (managed Airflow)
- **Visualization**: Looker Studio
- **Monitoring**: Cloud Monitoring

Implementation Details

1. Data Source & Ingestion

- Set up MySQL database with e-commerce schema:
  - `users` : Customer demographic information
  - `products` : Product catalog with details
  - `carts` : Shopping cart contents and status
  - `orders` : Completed transactions
- Configure Cloud Data Fusion for initial batch loading
- Implement CDC (Change Data Capture) using Debezium for incremental updates

2. Data Processing Layer

- Develop Dataflow pipelines for:
  - Data cleansing and standardization
  - Feature engineering (user engagement scores, product affinity)
  - Data enrichment (geographical information, temporal patterns)
- Use Dataproc for batch processing with PySpark:
  - Calculate performance metrics
  - Generate aggregation tables
  - Perform cohort analysis

3. Data Storage Layer

- Implement a multi-tier storage strategy:
  - **Raw Layer** (Cloud Storage): Original data in parquet format

    - **Curated Layer** (BigQuery): Processed analytics-ready datasets
    - **Serving Layer** (BigQuery): Business-specific views and aggregates
- Design BigQuery tables with proper partitioning and clustering

4. Analytics & Visualization

- Create Looker Studio dashboards for key business metrics:
    - **Sales Performance**: Revenue trends, conversion rates
    - **User Behavior**: Cart abandonment analysis, journey mapping
    - **Product Analytics**: Bestsellers, frequently bundled items
    - **Marketing Effectiveness**: Campaign attribution, promotion analysis

5. Orchestration & Monitoring

- Implement Cloud Composer (Airflow) DAGs for pipeline orchestration
- Configure automated data quality validations using Great Expectations
- Set up monitoring with Cloud Monitoring:
    - Pipeline execution metrics
    - Data freshness SLAs
    - Error rate alerting

6. Data Governance

- Implement column-level security for PII data
- Configure audit logging for data access
- Create data lineage documentation

Expected Outcomes

- End-to-end analytics platform with scheduled refreshes
- Self-service dashboards for business stakeholders
- Documented data dictionary and metric definitions
- Scalable architecture supporting growing data volumes

Technical Skills Demonstrated

- GCP service integration
- Batch and incremental data processing
- SQL and PySpark development
- Data warehouse design
- Pipeline orchestration
- Data visualization
- Performance optimization

Extension Possibilities

- Implement ML models for churn prediction or product recommendations
- Add real-time processing for instant cart analysis