As a Data Engineer building foundations, here are **40 practice questions** (10 per topic) specifically tailored to data engineering scenarios.

Topic 1: PRINT FUNCTION (Data Engineering Context)

1. Print a data ingestion log message: `"[INFO] Starting ETL process at 2024-01-15 08:00:00"`
2. Print a table schema in a readable format: column name, data type, and nullable status
3. Print a progress bar simulation: `"Processing: [##########] 100%"`
4. Print multiple data quality metrics in one line: `"Rows: 10000, Nulls: 23, Duplicates: 5"`
5. Print a SQL query with proper formatting (using triple quotes)
6. Print a CSV header row with 5 columns separated by pipes `|`
7. Print a warning message in yellow (simulate with text: `"WARNING: Missing timestamp column"`)
8. Print a dictionary of configuration parameters in a readable key-value format
9. Print a multi-line data pipeline DAG structure using `\n`
10. Print a data validation report with aligned columns

Topic 2: VARIABLES (Data Engineering Context)

1. Create variables to store database connection parameters: host, port, database, username
2. Create variables for a data schema: table_name (string), row_count (int), file_size_gb (float), is_partitioned (bool)
3. Create variables for three different data file paths and print them in a list format
4. Swap values between `source_system` and `target_system` variables
5. Create variables for data quality thresholds: null_threshold=0.05, duplicate_threshold=100, freshness_hours=24
6. Use chained assignment to set three different logging levels to the same boolean value `True`
7. Create a variable `full_table_name` by concatenating `schema_name` and `table_name` with a dot
8. Create variables for batch processing: `batch_size`, `current_batch`, `total_batches`
9. Store different data formats as variables: `csv_format`, `parquet_format`, `json_format`
10. Create variables for a data pipeline's start time, end time, and calculate duration

---

Topic 3: USER INPUT (Data Engineering Context)

1. Ask for a database table name and print a SELECT COUNT(*) query for it
2. Prompt for source and target file paths, then print a copy command
3. Ask for a column name and default value, then print an SQL COALESCE statement
4. Get a delimiter choice (comma, tab, pipe) and print a CSV reader configuration
5. Ask for data retention days and calculate the cutoff date
6. Prompt for environment (dev/stage/prod) and print the corresponding database host
7. Ask for a date in YYYY-MM-DD format and print a partition filter clause
8. Get a list of columns to select (comma-separated) and print a SELECT statement
9. Ask for error tolerance percentage and validate if it's between 0 and 100
10. Prompt for a file pattern (e.g., `*.csv`, `data_*.parquet`) and print a file search command

---

Topic 4: DATA TYPES (Data Engineering Context)

1. Check and print the data type of user input (could be string, number, or boolean)
2. Convert a string "2024-01-15" to a datetime object (simulate with string operations)
3. Create a list of column names from a comma-separated string
4. Convert a dictionary of string values to appropriate types: `{"rows": "1000", "size": "2.5", "active": "True"}`
5. Check if a value can be converted to integer for ID validation
6. Create a tuple for immutable configuration: database settings
7. Convert a list of strings to a set to find unique column values
8. Handle numeric strings with thousand separators: `"1,000,000"` → `1000000`
9. Create appropriate variables for different data types found in a CSV header
10. Convert boolean string values ("true"/"false", "yes"/"no", "1"/"0") to Python booleans