DATA ENGINEER ROADMAP

120 concepts to become a data engineer

8 areas, 15 concepts each. Every concept explained in one line, the order to study each area, and two practice questions to test yourself.

How to use it: one area at a time, tick each concept only when you can explain it out loud in under a minute, then answer the two practice questions without notes.

@chalkesahil · chalkesahil.ai

AREA 01 / 08

SQL

Study order: Write SQL every day; window functions and query plans come last.

01	**Joins**	Inner, left, full and when each drops or keeps rows.
02	**GROUP BY and HAVING**	Aggregate per group, filter after grouping.
03	**Window functions**	Running totals, ranks and comparisons without collapsing rows.
04	**CTEs**	Break long queries into readable steps.
05	**Subqueries**	Queries inside queries for filters and lookups.
06	**CASE WHEN**	Conditional logic inside a query.
07	**NULL handling**	COALESCE and why NULL breaks comparisons.
08	**Dedup with ROW_NUMBER**	Keep the latest record per key.
09	**Indexes**	Speed up reads, slow down writes.
10	**Query plans**	EXPLAIN shows where the time goes.
11	**Transactions**	All or nothing for a group of writes.
12	**Upserts and MERGE**	Insert new rows, update existing ones.
13	**Date and time functions**	Truncate, extract and handle time zones.
14	**Set operations**	UNION, INTERSECT, EXCEPT.
15	**SQL dialects**	Postgres, BigQuery and Snowflake differ in small ways.

Practice

1. Get each user's most recent order in one query.
2. A query takes 2 minutes. Walk through how you'd speed it up.

AREA 02 / 08

Python

Study order: Python for files, APIs and dataframes; you'll glue every tool with it.

01	**Core data types**	Lists, dicts, sets and when to use each.
02	**Functions and modules**	Reusable pieces for every pipeline step.
03	**File handling**	Read and write CSV, JSON and Parquet.
04	**Pandas**	Quick transforms on data that fits in memory.
05	**Polars**	Fast dataframe library for bigger local data.
06	**Calling APIs**	Pull data from REST APIs with pagination.
07	**Database connections**	SQLAlchemy or drivers to read and write tables.
08	**Generators**	Process large files without loading them all.
09	**Error handling**	Fail loudly with useful messages.
10	**Logging**	Know which record broke and why.
11	**Config management**	Environments and secrets outside code.
12	**Unit tests**	pytest for transform logic.
13	**Packaging**	Share code across pipelines as a package.
14	**Type hints**	Catch bugs before the pipeline runs.
15	**CLI scripts**	argparse or Typer for runnable jobs.

Practice

1. Pull all pages from a paginated API and save them as Parquet.
2. Process a 20 GB CSV on a laptop with 8 GB of RAM.

AREA 03 / 08

Data modeling

Study order: Model a real business, like an online store, end to end.

01	**OLTP vs OLAP**	App databases vs analytics databases.
02	**Normalisation**	Remove duplication in transactional tables.
03	**Denormalisation**	Duplicate on purpose to make analytics fast.
04	**Star schema**	Facts in the middle, dimensions around them.
05	**Fact tables**	Events and measurements, like orders.
06	**Dimension tables**	Context for facts, like customers and products.
07	**Grain**	What one row in a fact table means.
08	**Surrogate keys**	Warehouse-owned IDs instead of source IDs.
09	**Slowly changing dimensions**	Track history when a customer's city changes.
10	**Data vault**	Hubs, links and satellites for heavy-history models.
11	**Medallion architecture**	Bronze raw, silver cleaned, gold business-ready.
12	**Partitioning**	Split tables by date so queries scan less.
13	**Clustering**	Sort data so related rows sit together.
14	**Data contracts**	Agreed schemas between producers and consumers.
15	**Naming conventions**	Consistent names save hours of confusion.

Practice

1. Design the warehouse model for a food delivery app.
2. How would you track a customer's address history?

AREA 04 / 08

Spark

Study order: Learn the DataFrame API on a local cluster, then on Databricks.

01	**Why distributed**	Split work across machines when data is too big for one.
02	**Driver and executors**	Driver plans, executors do the work.
03	**RDD vs DataFrame**	Use DataFrames; RDDs are the low-level base.
04	**Lazy evaluation**	Nothing runs until an action like count or write.
05	**Transformations vs actions**	Build the plan, then trigger it.
06	**Partitions**	The unit of parallel work.
07	**Shuffles**	Data moving between machines; usually the slow part.
08	**Joins in Spark**	Broadcast small tables to avoid shuffles.
09	**Data skew**	One huge key makes one task run forever.
10	**Caching**	Keep reused data in memory.
11	**Spark SQL**	Write SQL on top of DataFrames.
12	**Parquet and Delta**	Columnar files; Delta adds transactions.
13	**PySpark UDFs**	Custom Python functions, slower than built-ins.
14	**Spark UI**	See stages, tasks and where time goes.
15	**Databricks**	Managed Spark widely used in Indian companies.

Practice

1. A Spark job is stuck at 199 of 200 tasks. What's happening?
2. Join a 1 TB table with a 50 MB table efficiently.

AREA 05 / 08

Airflow

Study order: Schedule one real pipeline end to end, with retries and alerts.

01	**Orchestration**	Run steps in the right order on a schedule.
02	**DAGs**	Tasks and their dependencies as a graph.
03	**Operators**	Ready-made task types for SQL, Python, cloud.
04	**Scheduling**	Cron expressions and data intervals.
05	**Backfills**	Re-run a pipeline for past dates.
06	**Retries**	Automatic re-runs on temporary failures.
07	**Idempotent tasks**	Running twice gives the same result.
08	**Sensors**	Wait for a file or table before starting.
09	**XComs**	Pass small values between tasks.
10	**Connections and secrets**	Store credentials outside the DAG code.
11	**Task failures and alerts**	Know within minutes when a pipeline breaks.
12	**SLAs**	Data must land by a promised time.
13	**dbt**	SQL transformations with tests and docs.
14	**dbt tests**	not_null, unique and relationships checks.
15	**Alternatives**	Prefect and Dagster solve the same problem.

Practice

1. Design a daily pipeline that loads yesterday's orders into the warehouse.
2. A DAG failed for 3 days. How do you recover the missing data?

AREA 06 / 08

Cloud warehouses

Study order: Pick one cloud and one warehouse; the ideas transfer.

01	**Data lake vs warehouse**	Cheap raw files vs fast structured tables.
02	**Lakehouse**	Warehouse features on top of lake files.
03	**Object storage**	S3, GCS or ADLS hold the raw data.
04	**Snowflake**	Separate storage and compute, pay per second.
05	**BigQuery**	Serverless warehouse billed by data scanned.
06	**Redshift**	AWS's warehouse; know sort and distribution keys.
07	**Columnar storage**	Read only the columns a query needs.
08	**ELT vs ETL**	Load raw first and transform inside the warehouse.
09	**Ingestion tools**	Fivetran or Airbyte to copy data from apps.
10	**Change data capture**	Stream database changes instead of full reloads.
11	**Cost control**	Partition, cluster and avoid SELECT *.
12	**Access control**	Roles and row-level security.
13	**Data catalog**	Find what data exists and who owns it.
14	**Data lineage**	Trace where each column came from.
15	**IAM basics**	Service accounts with least privilege.

Practice

1. Your BigQuery bill jumped 5x. How do you find the cause?
2. When would you choose a lakehouse over a warehouse?

AREA 07 / 08

Streaming and Kafka

Study order: Batch first; add streaming only when the business needs seconds.

01	**Batch vs streaming**	Run every hour vs process each event as it arrives.
02	**Kafka topics**	Named streams of events.
03	**Partitions**	How Kafka scales and keeps order per key.
04	**Producers and consumers**	Write events, read events.
05	**Consumer groups**	Share a topic's work across many readers.
06	**Offsets**	Where each consumer is in the stream.
07	**Delivery guarantees**	At most once, at least once, exactly once.
08	**Retention**	How long Kafka keeps events.
09	**Schema registry**	Keep producers and consumers agreeing on format.
10	**Spark Structured Streaming**	Streaming with the DataFrame API.
11	**Flink**	Low-latency stream processing.
12	**Windowing**	Group events into time windows.
13	**Late data and watermarks**	Handle events that arrive out of order.
14	**Dead letter queues**	Park bad events instead of blocking the stream.
15	**Backpressure**	What happens when consumers can't keep up.

Practice

1. Design a pipeline that shows live order counts per city.
2. Your consumer is lagging by 2 hours. What do you check?

AREA 08 / 08

AI data pipelines

Study order: This is where AI raised the demand: models are only as good as the data you feed them.

01	**Data quality checks**	Great Expectations or dbt tests on every load.
02	**Unstructured data**	PDFs, images and audio need different pipelines.
03	**Document parsing**	Extract clean text and tables from files.
04	**Chunking pipelines**	Split documents for RAG at scale.
05	**Embedding pipelines**	Batch-generate vectors and keep them updated.
06	**Vector databases**	pgvector, Pinecone or Weaviate for retrieval.
07	**Feature stores**	Serve the same features for training and prediction.
08	**Training datasets**	Versioned, labelled data for models.
09	**Data labelling**	Human and model-assisted labelling workflows.
10	**Data versioning**	Delta time travel or DVC to reproduce a dataset.
11	**PII masking**	Remove personal data before it reaches a model.
12	**Freshness for AI**	Stale data means confidently wrong answers.
13	**LLM-powered transforms**	Use models to classify or extract fields at scale.
14	**Cost of AI pipelines**	Embedding and model calls cost money per row.
15	**Portfolio project**	One pipeline from raw files to a working RAG index.

Practice

1. Build a pipeline that keeps a RAG index in sync with a folder of PDFs.
2. How would you stop customer phone numbers from reaching an LLM?