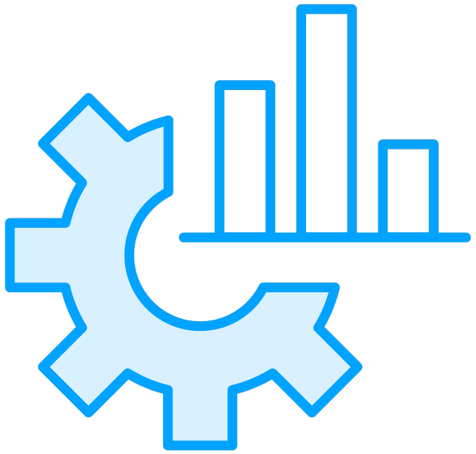
Handling Large Queries and Cloud-hosted Databases with Python

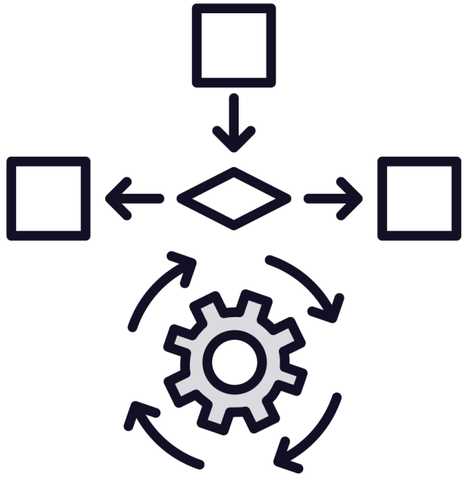
Pinal Dave
Data Expert

@pinaldave | blog.sqlauthority.com

Why Use Cloud-hosted Databases?



Scalability, high availability, and managed infrastructure



Big data processing, analytics, and distributed querying

Native libraries for seamless integration

Popular Python Libraries for Cloud Databases

Library	Cloud database	Purpose
google-cloud-bigquery	Google BigQuery	Query execution, data analysis
snowflake-connector-python	Snowflake	Secure connection, data retrieval
psycopg2 / SQLAlchemy	Amazon Redshift	Query execution, ORM support

Steps to Working with Cloud Databases in Python

Connect to Google BigQuery, Snowflake, or Redshift

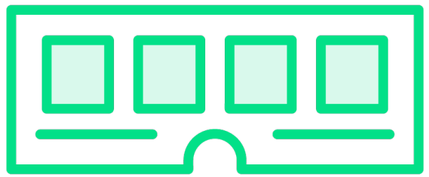
Load cloud data into Pandas DataFrame

Analyze, filter, and transform data seamlessly

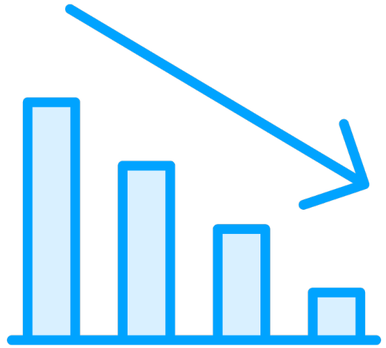
```
from google.cloud import bigquery

client = bigquery.Client()
query = "SELECT name, age FROM my_dataset.users LIMIT 10"
result = client.query(query).to_dataframe()

print(result)
```

Connecting to Google BigQuery

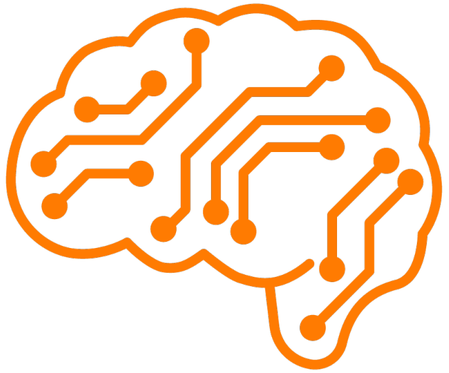
Challenges with Large Datasets



Processing large datasets can exhaust system memory



Performance degrades with inefficient data loading



Efficient techniques reduce RAM usage while processing big data

Techniques for Handling Large Data

Method	Use case	Example
Lazy loading	Load data only when needed	Python generators
Chunk processing	Process data in batches	Pandas (read_csv(chunk_size=...))
Dask processing	Parallel computation for big data	dask.dataframe

Why Security Matters in Cloud Databases

Data breaches due to misconfigured access

Weak authentication can expose sensitive data

Unoptimized queries increase cloud costs

Best Practices for Secure Querying

Security measure	Purpose	Example
IAM roles & permissions	Control who can access what	AWS IAM policies
Environment variables	Secure database credentials	os.environ.get("DB_PASSWORD")
Encryption	Protect data in transit and at rest	TLS, KMS, or column-level encryption

Secure Querying in Snowflake

```
import snowflake.connector
import os

conn = snowflake.connector.connect(
    user=os.environ['SNOWFLAKE_USER'],
    password=os.environ['SNOWFLAKE_PASSWORD'],
    account="my_account"
)
cur = conn.cursor()
cur.execute("SELECT * FROM sales_data LIMIT 100")
result = cur.fetchall()
```

Thank you for your attention

pinal@sqlauthority.com | https://blog.sqlauthority.com