Database Operations

# Introduction
The database service in Supabase is a managed PostgreSQL instance. It offers SQL capabilities, auto-generated APIs, and real-time data updates.

# Connecting to the Database
The Python client interacts with the database using REST interfaces automatically generated for your tables.

Example:
```
data = supabase.table("users").select("*").execute()
print(data)
```

# Inserting Data
```
new_user = {"name": "John", "age": 30}
supabase.table("users").insert(new_user).execute()
```

# Updating Data
```
supabase.table("users").update({"age": 31}).eq("name", "John").execute()
```

# Deleting Data
```
supabase.table("users").delete().eq("name", "John").execute()
```

# Real-time Queries
Supabase allows you to subscribe to real-time changes:
```
def on_update(payload):
print("Update:", payload)
supabase.table("messages").on("UPDATE", on_update).subscribe()
```

# SQL Editor
In addition to the Python client, Supabase provides an in-browser SQL editor to run custom queries, manage schemas, and view results.

# Relationships
Supabase fully supports foreign keys, joins, and constraints — everything you’d expect from PostgreSQL.

# Performance Tips
- Index frequently queried columns.
- Use pagination for large datasets.
- Limit data fetches to avoid unnecessary payloads.

# Summary
Supabase provides a powerful, fully managed PostgreSQL database accessible via REST or real-time APIs — combining the reliability of SQL with the ease of modern backend tools.