Mindmap to Learn PostgreSQL

* **Introduction to PostgreSQL**
  + **Overview**: PostgreSQL is an advanced, open-source relational database management system (RDBMS) known for its robustness, scalability, and support for SQL standards.
  + **Installation**: Install PostgreSQL on various platforms (Windows, macOS, Linux) using package managers or installer downloads.
* **Basic Concepts**
  + **Database**: A collection of organized data.
    - Example: CREATE DATABASE mydatabase;.
  + **Schema**: A namespace within a database to organize objects.
    - Example: CREATE SCHEMA myschema;.
  + **Table**: A collection of rows and columns to store data.
    - Example: CREATE TABLE users (id SERIAL PRIMARY KEY, name VARCHAR(100));.
* **Data Types**
  + **Numeric Types**: Integer, Float, Serial, etc.
    - Example: INTEGER, SERIAL, FLOAT.
  + **String Types**: CHAR, VARCHAR, TEXT.
    - Example: VARCHAR(255), TEXT.
  + **Date/Time Types**: DATE, TIME, TIMESTAMP.
    - Example: TIMESTAMP.
  + **Boolean Types**: BOOLEAN.
    - Example: BOOLEAN.
  + **Composite Types**: Arrays, JSON, JSONB.
    - Example: INTEGER[], JSON, JSONB.
* **SQL Basics**
  + **CRUD Operations**: Create, Read, Update, Delete.
    - **Create**: INSERT INTO users (name) VALUES ('John Doe');.
    - **Read**: SELECT \* FROM users;.
    - **Update**: UPDATE users SET name = 'Jane Doe' WHERE id = 1;.
    - **Delete**: DELETE FROM users WHERE id = 1;.
  + **Basic Queries**: Select statements with conditions.
    - Example: SELECT \* FROM users WHERE name = 'John Doe';.
  + **Joins**: Combine rows from two or more tables.
    - Example: SELECT users.name, orders.amount FROM users JOIN orders ON users.id = orders.user\_id;.
* **Indexes**
  + **Creating Indexes**: Improve the speed of data retrieval.
    - Example: CREATE INDEX idx\_users\_name ON users (name);.
  + **Types of Indexes**: B-tree, Hash, GiST, GIN, BRIN.
    - Example: CREATE INDEX idx\_users\_name ON users USING HASH (name);.
* **Constraints**
  + **Primary Key**: Uniquely identifies each row.
    - Example: PRIMARY KEY (id).
  + **Foreign Key**: Enforces a link between the data in two tables.
    - Example: FOREIGN KEY (user\_id) REFERENCES users(id).
  + **Unique**: Ensures all values in a column are unique.
    - Example: UNIQUE (email).
  + **Check**: Ensures the validity of data based on a condition.
    - Example: CHECK (age >= 18).
* **Advanced SQL**
  + **Subqueries**: Nested queries within another query.
    - Example: SELECT \* FROM users WHERE id IN (SELECT user\_id FROM orders);.
  + **Common Table Expressions (CTE)**: Temporary result set.
    - Example: WITH cte AS (SELECT \* FROM users) SELECT \* FROM cte;.
  + **Window Functions**: Perform calculations across a set of table rows.
    - Example: SELECT name, SUM(amount) OVER (PARTITION BY name) FROM orders;.
  + **Transactions**: Ensure data integrity.
    - Example: BEGIN; UPDATE users SET name = 'John'; COMMIT;.
* **Functions and Procedures**
  + **User-defined Functions**: Create custom functions.
    - Example: CREATE FUNCTION get\_user\_fullname() RETURNS TEXT AS $$ BEGIN RETURN 'John Doe'; END; $$ LANGUAGE plpgsql;.
  + **Stored Procedures**: Encapsulate business logic.
    - Example: CREATE PROCEDURE update\_user\_name(IN user\_id INT, IN new\_name TEXT) LANGUAGE plpgsql AS $$ BEGIN UPDATE users SET name = new\_name WHERE id = user\_id; END; $$;.
* **Triggers**
  + **Creating Triggers**: Automatically execute a function.
    - Example: CREATE TRIGGER update\_timestamp BEFORE UPDATE ON users FOR EACH ROW EXECUTE FUNCTION update\_timestamp();.
  + **Trigger Functions**: Functions invoked by triggers.
    - Example: CREATE FUNCTION update\_timestamp() RETURNS TRIGGER AS $$ BEGIN NEW.updated\_at = NOW(); RETURN NEW; END; $$ LANGUAGE plpgsql;.
* **Performance Optimization**
  + **Query Optimization**: Use EXPLAIN to analyze query performance.
    - Example: EXPLAIN ANALYZE SELECT \* FROM users;.
  + **Vacuum**: Reclaim storage and optimize database performance.
    - Example: VACUUM ANALYZE;.
  + **Connection Pooling**: Use tools like PgBouncer to manage database connections.
* **Security**
  + **Roles and Permissions**: Manage database access.
    - Example: CREATE ROLE readonly; GRANT SELECT ON ALL TABLES IN SCHEMA public TO readonly;.
  + **Authentication**: Use methods like MD5, SCRAM-SHA-256 for securing connections.
  + **Encryption**: Encrypt data in transit and at rest.
* **Backup and Restore**
  + **Backup**: Use pg\_dump to create backups.
    - Example: pg\_dump mydatabase > mydatabase.sql.
  + **Restore**: Use psql to restore from backups.
    - Example: psql mydatabase < mydatabase.sql.
* **Replication and High Availability**
  + **Streaming Replication**: Set up primary and standby servers.
    - Example: Configure primary\_conninfo in recovery.conf.
  + **Logical Replication**: Replicate data between databases.
    - Example: CREATE PUBLICATION mypublication FOR TABLE mytable;.
  + **Failover and Load Balancing**: Use tools like Patroni or repmgr.
* **Extensions**
  + **PostGIS**: Spatial and geographic objects for PostgreSQL.
    - Example: CREATE EXTENSION postgis;.
  + **HStore**: Key-value store within PostgreSQL.
    - Example: CREATE EXTENSION hstore;.
  + **Full-Text Search**: Advanced search capabilities.
    - Example: CREATE EXTENSION unaccent;.
* **Data Import and Export**
  + **CSV Import/Export**: Load and dump data using CSV format.
    - Import: COPY users FROM 'file.csv' DELIMITER ',' CSV HEADER;.
    - Export: COPY users TO 'file.csv' DELIMITER ',' CSV HEADER;.
  + **Foreign Data Wrappers**: Access external data sources.
    - Example: CREATE EXTENSION postgres\_fdw; CREATE SERVER foreign\_server FOREIGN DATA WRAPPER postgres\_fdw OPTIONS (host 'hostname', dbname 'foreign\_db');.
* **Monitoring and Maintenance**
  + **pg\_stat\_statements**: Track execution statistics of all SQL statements.
    - Example: CREATE EXTENSION pg\_stat\_statements;.
  + **pgAdmin**: GUI tool for managing PostgreSQL databases.
  + **AutoVacuum**: Automatically vacuum and analyze tables.
* **Development Tools**
  + **psql**: Command-line interface for PostgreSQL.
  + **pgAdmin**: Graphical user interface for managing PostgreSQL.
  + **DBeaver**: Universal database tool for developers and database administrators.
* **Best Practices**
  + **Regular Backups**: Schedule regular backups and test recovery.
    - Example: Use cron jobs for automated backups.
  + **Indexing Strategy**: Use indexes wisely to optimize queries.
  + **Schema Design**: Normalize database schema to reduce redundancy and improve integrity.