PostgreSQL Comprehensive Revision Notes

📚 Resources

These materials support a full-length PostgreSQL tutorial covering database fundamentals, SQL syntax, constraints, relationships, and advanced relational concepts.

- 🌐 Official PostgreSQL Website
- 📘 W3Schools PostgreSQL Tutorial
- 🎥 Full PostgreSQL YouTube Tutorial
- 🔗 Video Tutorial Notes Source (bit.ly redirect)

📝 *These notes were taken using Obsidian for better structure, revision, and future reference.*

You can paste this section at the beginning of your note to give it a clean, professional touch. Want a footer too (with a personal tag like "Made with 💻 by Maryam Osman")?

📌 Introduction to PostgreSQL

– Overview

- PostgreSQL is a powerful open-source relational database.
- Widely used in startups and enterprises.
- Preferred for performance, flexibility, and SQL support.

Key Ideas:

- Learn PostgreSQL using terminal (not GUI).
- PostgreSQL = RDBMS + SQL (Structured Query Language).
- Tables, rows, and relationships form the core.

🖥️ Installation & Setup

- Install PostgreSQL on **Windows or Mac**.
- Choose a GUI (like PGAdmin) or terminal-based client.

Connection Methods:

- Command line (`psql`)
- GUI Clients (Postico, PGAdmin)
- Server-side apps (e.g., Python scripts)

🛠️ Working with Databases

- Create, connect to, and delete databases.
- Learn the `DROP DATABASE` warning: data loss is irreversible.

```
CREATE DATABASE mydb;
\c mydb
DROP DATABASE mydb;
```

🧱 Table Creation & Constraints

- Define table schemas using `CREATE TABLE`.
- Add **data types**: `INT`, `VARCHAR`, `DATE`, etc.
- Add constraints: `NOT NULL`, `PRIMARY KEY`

```
CREATE TABLE people (
  id SERIAL PRIMARY KEY,
  name VARCHAR(50) NOT NULL,
  age INT CHECK (age > 0)
);
```

📥 Inserting & Managing Data

- Use `INSERT INTO` for data.

- Mockaroo to generate 1000+ rows.

```
INSERT INTO people (name, age) VALUES ('Maryam', 22);
```

🔍 Querying Data

- Filtering with `WHERE`, `AND`, `OR`
- Sorting with `ORDER BY`
- Remove duplicates using `DISTINCT`
- Limiting results with `LIMIT`, `FETCH`, `BETWEEN`, `LIKE`, `ILIKE`

```
SELECT * FROM people WHERE age > 20 ORDER BY name;
```

📊 Aggregations & Grouping

- `GROUP BY`, `HAVING`, `COUNT`, `MAX`, `MIN`, `AVG`, `SUM`
- Use HAVING to filter grouped results.

```
SELECT country, COUNT(*) FROM users GROUP BY country HAVING COUNT(*) > 100;
```

➕ Arithmetic & Null Safety

- Use `+`, `-`, `*`, `/`, `%`, `ROUND`
- Prevent division by zero: `NULLIF`, `COALESCE`

```
SELECT price, price * 0.8 AS discounted_price FROM products;
```

📅 Working with Dates

- Subtract/add dates, extract fields, use `AGE()`

```
SELECT AGE(birthdate) FROM people;
```

🔑 Keys and Constraints

2:31:56–2:49:15

- **Primary Keys** ensure unique rows
- **Unique Constraints** prevent duplicate values
- **Check Constraints** enforce value rules

```
ALTER TABLE users ADD CONSTRAINT unique_email UNIQUE(email);
```

✏️ Updating & Conflict Handling

- `UPDATE`, `ON CONFLICT`, `UPSERT`

```
INSERT INTO users (id, name) VALUES (1, 'Maryam')
ON CONFLICT (id) DO UPDATE SET name = EXCLUDED.name;
```

🔗 Foreign Keys & Relationships

- Create one-to-one, one-to-many links
- Ensure matching foreign key values

```
ALTER TABLE cars ADD CONSTRAINT fk_owner FOREIGN KEY (owner_id) REFERENCES
people(id);
```

📤 Exporting & Extensions

- Export data to CSV using `COPY` command
- Use `SERIAL`, `SEQUENCES`, `UUID` for unique IDs

```sql
COPY users TO '/tmp/users.csv' DELIMITER ',' CSV HEADER;
```

🔐 UUIDs as Primary Keys

- Prevent predictable ID sequences
- Useful for distributed systems

```sql
CREATE TABLE users (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name TEXT
);
```

✅ Final Notes

- Mastering PostgreSQL is essential for data engineers.
- Focus on: **DDL**, **DML**, **Joins**, **Keys**, **Constraints**, **Aggregate Functions**.
- Practice using **terminal** + **SQL scripts** to simulate real projects.
