

Class #11: Introduction to SQL - Part 1

(Data Scraping)

5. Filtering data using the WHERE clause and combining conditions.
6. Sorting data with keywords such as DISTINCT, TOP, LIKE, etc.
7. Modifying data using SQL syntax.

Database Operations:

মূল টেবিলকে পরিবর্তন করা। মূল টেবিলের গঠন পরিবর্তন করা যায়, নিম্নোক্ত ৮ ভাবেঃ

1. PostgreSQL CREATE TABLE

PostgreSQL CREATE TABLE: ডেটাবেজে নতুন একটি টেবিল তৈরি করতে ব্যবহৃত হয়। গত লেকচারে আমরা এই ব্যাপার জেনেছিলাম।

```
CREATE TABLE table_name (
    'column1' datatype,
    'column2' datatype,
    ...
);
```

ব্যাখ্যা:

1. CREATE TABLE:
 - এটি একটি SQL কমান্ড যা নতুন একটি টেবিল তৈরি করতে ব্যবহৃত হয়।
2. table_name:
 - এখানে আপনি যে টেবিলটি তৈরি করতে চান তার নাম লিখবেন। টেবিলের নামটি বেছে নেওয়ার সময় কিছু নিয়ম অনুসরণ করতে হবে, যেমন স্পেস ব্যবহার করা যাবে না এবং এটি ইউনিক হতে হবে।
3. 'column1', 'column2', ...:
 - এগুলো টেবিলের কলাম (column) এর নাম। প্রতিটি কলামের জন্য একটি নাম নির্ধারণ করা হয়। কলামগুলোর নামও কিছু নিয়ম অনুসরণ করতে হয়, যেমন স্পেস ব্যবহার করা যাবে না, নামটি সহজ এবং অর্থপূর্ণ হওয়া উচিত ইত্যাদি।
4. datatype:
 - কলামের জন্য যে ধরনের ডেটা রাখা হবে, তার ধরন নির্ধারণ করা হয়। এটি যেমন হতে পারে:
 - INT: পূর্ণসংখ্যা (integer)
 - VARCHAR(n): স্ট্রিং বা টেক্স্ট, যেখানে n হলো স্ট্রিংয়ের সর্বোচ্চ দৈর্ঘ্য
 - DATE: তারিখ
 - FLOAT: দশমিক সংখ্যা

 BOOLEAN: সত্য বা মিথ্যা (True/False)

উদাহরণস্বরূপ:

- column1 হতে পারে VARCHAR(255) যা একটি টেক্সট কলাম, এবং column2 হতে পারে INT যা একটি পূর্ণসংখ্যা কলাম।

উদাহরণ:

```
CREATE TABLE Employees (
    employee_id INT,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    hire_date DATE
);
```

এখানে:

-  Employees হলো টেবিলের নাম।
-  employee_id, first_name, last_name, hire_date হলো কলামের নাম।
-  INT, VARCHAR(50), DATE হলো তাদের ডেটাটাইপ।

এই SQL কমান্ডটি এক নতুন Employees টেবিল তৈরি করবে, যেখানে কর্মচারীর আইডি, প্রথম নাম, শেষ নাম এবং নিয়োগ তারিখ থাকবে।

সারাংশ:

- CREATE TABLE স্টেটমেন্টের মাধ্যমে আপনি একটি নতুন টেবিল তৈরি করতে পারেন।
- আপনি টেবিলের নাম এবং কলামগুলোর নাম ও ডেটাটাইপ নির্ধারণ করবেন।

2. PostgreSQL INSERT INTO

PostgreSQL INSERT INTO: টেবিলে নতুন ডাটা যোগ করতে ব্যবহৃত হয়। টেবিল তৈরি করলে, টেবিল খালি থাকে। তাতে ডেটা যুক্ত করার প্রয়োজন হয়। তখন, insert into দিয়ে ডেটা যোগ করা যায়। এটি একটি টেবিলে ডেটা ইনসার্ট (বা সংযোজন) করার জন্য ব্যবহৃত হয়।

```
INSERT INTO table_name (column1, column2, ...)
VALUES
(value1, value2, ...),
(value11, value12, ...),
(value21, value22, ...),
(value31, value32, ...),
(value41, value42, ...);
```

ব্যাখ্যা:

1. INSERT INTO:
 - এটি একটি SQL কমান্ড যা একটি টেবিলে নতুন ডেটা (বা রেকর্ড) ইনসার্ট করতে ব্যবহৃত হয়।
2. table_name:

- এখানে আপনি যে টেবিলে ডেটা ইনসার্ট করতে চান তার নাম দিবেন। উদাহরণস্বরূপ, যদি টেবিলটির নাম Employees হয়, তাহলে এখানে Employees লিখবেন।

3. (column1, column2, ...):

- এগুলো হলো সেই কলামগুলো যেগুলোর মধ্যে আপনি ডেটা ইনসার্ট করবেন। কলামগুলোর নাম সঠিকভাবে উল্লেখ করতে হবে, কারণ প্রতিটি ভ্যালু ঠিক একটি নির্দিষ্ট কলামে ইনসার্ট হবে।
- কলামগুলো সুনির্দিষ্ট করা বাধ্যতামূলক নয়, তবে যদি কলামগুলো উল্লেখ করা না হয়, তবে টেবিলের সকল কলামের জন্য ডেটা প্রদান করতে হবে।

4. VALUES:

- VALUES কীওয়ার্ডটি টেবিলে ডেটা ইনসার্ট করার জন্য ব্যবহৃত হয়। এর পরে আপনি যেসব মান ইনপুট দিতে চান সেগুলো উল্লেখ করবেন।

5. (value1, value2, ...), (value11, value12, ...), ...:

- প্রতিটি প্যারেনথেসিসে () একটি রেকর্ডের মান রয়েছে, যেখানে প্রতিটি মান একটি নির্দিষ্ট কলামের জন্য।
- প্রতিটি রেকর্ড (বা পংক্তি) আলাদা আলাদা প্যারেনথেসিসে রাখা হয় এবং কমা দিয়ে পৃথক করা হয়।

উদাহরণ:

ধরা যাক, আপনার একটি Employees টেবিল আছে, যার তিনটি কলাম: employee_id, first_name, এবং last_name। এখন আপনি কিছু কর্মচারীর তথ্য ইনসার্ট করতে চান।

```
INSERT INTO Employees (employee_id, first_name, last_name)
VALUES
(1, 'John', 'Doe'),
(2, 'Jane', 'Smith'),
(3, 'Mike', 'Johnson');
```

এখানে:

- Employees: টেবিলের নাম।
- employee_id, first_name, last_name: কলামের নাম।
- VALUES (1, 'John', 'Doe'), (2, 'Jane', 'Smith'), (3, 'Mike', 'Johnson'): আপনি যে মানগুলো ইনসার্ট করছেন, সেগুলো হল তিনটি ভিন্ন রেকর্ড:
 - প্রথম রেকর্ড: employee_id = 1, first_name = 'John', last_name = 'Doe'
 - দ্বিতীয় রেকর্ড: employee_id = 2, first_name = 'Jane', last_name = 'Smith'
 - তৃতীয় রেকর্ড: employee_id = 3, first_name = 'Mike', last_name = 'Johnson'

আরেকটি উদাহরণ (বেশি কলাম এবং মান):

```
INSERT INTO Products (product_id, product_name, price, quantity)
VALUES
(101, 'Laptop', 50000, 20),
(102, 'Mobile Phone', 15000, 50),
(103, 'Headphones', 3000, 100);
```

এখানে:

- Products টেবিলের মধ্যে তিনটি রেকর্ড ইনসার্ট হচ্ছে। প্রতিটি রেকর্ডে product_id, product_name, price, এবং quantity কলামের মান দেওয়া হচ্ছে।

সারাংশ:

- INSERT INTO স্টেটমেন্ট দিয়ে টেবিলে ডেটা ইনসার্ট করা হয়।
- আপনি যেসব কলামে ডেটা ইনসার্ট করবেন, সেগুলোর নাম নির্দিষ্ট করেন, এবং তারপর VALUES কীওয়ার্ডের মাধ্যমে প্রতিটি রেকর্ডের মান উল্লেখ করেন।
- একাধিক রেকর্ড একসঙ্গে ইনসার্ট করার জন্য VALUES প্যারেনথেসিসের মধ্যে কমা দিয়ে আলাদা করা হয়।

3. PostgreSQL ADD COLUMN

PostgreSQL ADD COLUMN: একটি টেবিলের মধ্যে নতুন অতিরিক্ত কলাম (column) যোগ করার জন্য ব্যবহৃত হয়। alter keyword ব্যবহার করতে হয়, টেবিলের নামের পূর্বে।

```
ALTER TABLE table_name
ADD COLUMN new_column datatype;
```

ব্যাখ্যা:

1. ALTER TABLE:

- ALTER TABLE হলো SQL কমাণ্ড, যা একটি বিদ্যমান টেবিলের গঠন পরিবর্তন করতে ব্যবহৃত হয়। এর মাধ্যমে আপনি টেবিলের কলাম যোগ, মুছে ফেলা বা পরিবর্তন করতে পারেন।

2. table_name:

- এটি টেবিলটির নাম যা আপনি পরিবর্তন করতে চান। উদাহরণস্বরূপ, যদি টেবিলটির নাম Employees হয়, তাহলে এখানে Employees লিখবেন।

3. ADD COLUMN:

- ADD COLUMN নির্দেশ করে যে আপনি নতুন একটি কলাম টেবিলে যোগ করতে চান।

4. new_column:

- এটি নতুন কলামের নাম। আপনি এখানে যে কলামটি যোগ করতে চান তার নাম দেবেন।

5. datatype:

- এখানে আপনি নতুন কলামের জন্য ডেটাটাইপ উল্লেখ করবেন। ডেটাটাইপ হতে পারে যেমন: INT (পুরো সংখ্যা), VARCHAR(n) (টেক্সট স্ট্রিং), DATE (তারিখ), FLOAT (দশমিক সংখ্যা) ইত্যাদি। ডেটাটাইপের মাধ্যমে আপনি কলামে কী ধরনের ডেটা রাখতে চান, সেটি নির্ধারণ করবেন।

উদাহরণ:

ধরা যাক, আপনার একটি Employees টেবিল আছে, যার বর্তমান কলামগুলো হলো employee_id, first_name, এবং last_name। এখন আপনি একটি নতুন কলাম hire_date যোগ করতে চান, যেখানে কর্মচারীর নিয়োগ তারিখ থাকবে।

```
ALTER TABLE Employees
ADD COLUMN hire_date DATE;
```

এখানে:

- ALTER TABLE Employees: Employees টেবিলের গঠন পরিবর্তন করতে বলছে।

- ADD COLUMN hire_date DATE: নতুন কলাম hire_date যোগ করা হচ্ছে, যার ডেটাটাইপ DATE (তারিখ)।

সারাংশ:

- ALTER TABLE কমান্ড দিয়ে আপনি একটি বিদ্যমান টেবিলের গঠন পরিবর্তন করতে পারেন।
- ADD COLUMN দিয়ে আপনি টেবিলে একটি নতুন কলাম যোগ করতে পারেন।
- নতুন কলামের নাম এবং ডেটাটাইপ নির্ধারণ করতে হবে, যেমন new_column datatype।

এটি টেবিলের গঠন পরিবর্তন করার একটি গুরুত্বপূর্ণ উপায়, যা আপনি প্রয়োজনে ডেটাবেস স্কিমা পরিবর্তন করতে ব্যবহার করতে পারেন।

4. PostgreSQL UPDATE

PostgreSQL UPDATE: UPDATE স্টেটমেন্টে, যা একটি টেবিলের নির্দিষ্ট রেকর্ডগুলির মান আপডেট (অথবা পরিবর্তন) করার জন্য ব্যবহৃত হয়।

```
UPDATE table_name
SET column1 = value1, column2 = value2
WHERE condition;
```

ব্যাখ্যা:

1. UPDATE:

- UPDATE SQL কমান্ডটি একটি টেবিলের বিদ্যমান ডেটা পরিবর্তন (অথবা আপডেট) করতে ব্যবহৃত হয়। এটি সেই রেকর্ড বা সারির মান পরিবর্তন করে যেখানে কিছু শর্ত (condition) পূর্ণ হয়।

2. table_name:

- এটি সেই টেবিলের নাম যেখানে আপনি ডেটা আপডেট করতে চান। উদাহরণস্বরূপ, যদি টেবিলটির নাম Employees হয়, তাহলে এখানে Employees লিখবেন।

3. SET:

- SET কীওয়ার্ডটি ব্যবহার করা হয় যাতে নির্দিষ্ট কলামের মান আপডেট করা যায়। এটি আপনার নির্দিষ্ট কলামগুলোর জন্য নতুন মান প্রদান করতে সাহায্য করে।

4. column1 = value1, column2 = value2:

- এখানে column1, column2 হল সেই কলামগুলির নাম যা আপনি আপডেট করতে চান, এবং value1, value2 হল নতুন মান যা আপনি ওই কলামগুলিতে সেট করতে চান।
- আপনি একাধিক কলামের মান একসঙ্গে আপডেট করতে পারেন, যার জন্য প্রতিটি কলাম এবং মানের মধ্যে কমা (,) ব্যবহার করতে হবে।

5. WHERE condition:

- WHERE কীওয়ার্ডটি একটি শর্ত (condition) নির্ধারণ করতে ব্যবহৃত হয়, যার মাধ্যমে নির্দিষ্ট রেকর্ডগুলো চিহ্নিত করা হয় যেগুলোর মান আপডেট করতে হবে।
- WHERE ছাড়া যদি কমান্ডটি চালানো হয়, তাহলে টেবিলের সব রেকর্ড আপডেট হয়ে যাবে। এজন্য WHERE শর্ত দেওয়া খুবই গুরুত্বপূর্ণ। উদাহরণস্বরূপ, WHERE employee_id = 1 নির্দেশ করে যে কেবল employee_id 1 এর রেকর্ডটিই আপডেট হবে।

উদাহরণ 1:

ধরা যাক, আপনার একটি Employees টেবিল আছে যার কলামগুলো হলো employee_id, first_name, last_name, এবং hire_date। এখন, আপনি employee_id = 1 এর জন্য first_name এবং last_name আপডেট করতে চান।

```
UPDATE Employees
```

```
SET first_name = 'John', last_name = 'Smith'
```

```
WHERE employee_id = 1;
```

এখানে:

- UPDATE Employees: Employees টেবিলটিতে ডেটা আপডেট করা হবে।
- SET first_name = 'John', last_name = 'Smith': first_name কলামের মান 'John' এবং last_name কলামের মান 'Smith' করা হবে।
- WHERE employee_id = 1: কেবল employee_id = 1 এর রেকর্ডে এই পরিবর্তন হবে।

উদাহরণ 2:

ধরা যাক, আপনি Products টেবিলের price কলামের মান আপডেট করতে চান যেখানে পণ্যের product_id 101 এবং 102।

```
UPDATE Products
```

```
SET price = 55000
```

```
WHERE product_id IN (101, 102);
```

এখানে:

- UPDATE Products: Products টেবিলের ডেটা আপডেট হবে।
- SET price = 55000: price কলামের মান 55000 করা হবে।
- WHERE product_id IN (101, 102): কেবল product_id 101 এবং 102 এর রেকর্ডগুলোর দাম পরিবর্তিত হবে।

উদাহরণ 3:

ধরা যাক, আপনি সমস্ত Employees যারা 2020 সালে নিয়োগ পেয়েছেন, তাদের status কলামকে "active" হিসেবে আপডেট করতে চান।

```
UPDATE Employees
```

```
SET status = 'active'
```

```
WHERE hire_date >= '2020-01-01';
```

এখানে:

- SET status = 'active': status কলামের মান 'active' করা হবে।
- WHERE hire_date >= '2020-01-01': কেবল সেই সব রেকর্ডগুলোর hire_date 2020 সালের 1 জানুয়ারির পরে হওয়া রেকর্ডগুলো আপডেট হবে।

সারাংশ:

- UPDATE কমান্ড টেবিলের রেকর্ডের মান পরিবর্তন করতে ব্যবহৃত হয়।
- SET কীওয়ার্ডের মাধ্যমে আপনি যেসব কলামের মান আপডেট করতে চান, সেগুলোর নাম এবং নতুন মান প্রদান করেন।

- WHERE শর্ত দ্বারা আপনি নির্দিষ্ট রেকর্ডগুলি চিহ্নিত করেন যেগুলোর মান পরিবর্তন করতে চান। WHERE শর্ত ছাড়া আপডেট করলে সমস্ত রেকর্ড আপডেট হয়ে যাবে, যা সাধারণত এড়ানো উচিত।

5. PostgreSQL ALTER COLUMN

PostgreSQL ALTER COLUMN: ALTER TABLE SQL কমান্ডটি একটি টেবিলের কাঠামো পরিবর্তন করার জন্য ব্যবহার করা হয়। যখন আপনি একটি কলামের ডেটাটাইপ পরিবর্তন করতে চান, তখন আপনি ALTER COLUMN ব্যবহার করেন। এর মাধ্যমে আপনি একটি কলামের ডেটাটাইপ পরিবর্তন করতে পারেন, যেমন সংখ্যা থেকে স্ট্রিং।

```
ALTER TABLE table_name
ALTER COLUMN column_name SET DATA TYPE new_datatype;
```

যখন আপনি একটি কলামের ডেটা টাইপ পরিবর্তন করতে চান, তখন আপনি ALTER COLUMN এবং SET DATA TYPE ব্যবহার করেন।

এখানে:

- table_name: আপনার টেবিলের নাম
- column_name: সেই কলামের নাম যার ডেটা টাইপ পরিবর্তন করতে চান
- new_datatype: নতুন ডেটা টাইপ যা আপনি কলামে সেট করতে চান

উদাহরণ:

ধরা যাক, আপনার একটি টেবিল আছে যার নাম students, এবং সেখানে একটি কলাম আছে যার নাম age, যার ডেটা টাইপ INTEGER। আপনি এই age কলামের ডেটা টাইপ পরিবর্তন করে VARCHAR করতে চান (যাতে বয়সটি টেক্স্ট আকারে সংরক্ষিত থাকে)।

এখন, SQL কমান্ডটি হবে:

```
ALTER TABLE students
ALTER COLUMN age SET DATA TYPE VARCHAR(10);
```

এটি age কলামের ডেটা টাইপকে VARCHAR(10) এ পরিবর্তন করবে, অর্থাৎ, এটি এখন 10টি অক্ষর পর্যন্ত টেক্স্ট ধারণ করতে পারবে।

মনে রাখবেন:

- কিছু ডেটা টাইপ পরিবর্তন করার সময় (যেমন, INTEGER থেকে VARCHAR বা DATE থেকে TEXT) ডেটা হারানোর ঝুঁকি থাকতে পারে। তাই আগে ডেটা ব্যাকআপ নেওয়া গুরুত্বপূর্ণ।
- কখনও কখনও SQL কমান্ডটি সফলভাবে চালাতে হলে আপনাকে টেবিলের মধ্যে কিছু ডেটা পরিবর্তন করতে হতে পারে (যেমন, যদি নতুন ডেটা টাইপ পুরানো ডেটার সাথে সামঞ্জস্যপূর্ণ না হয়)।

6. PostgreSQL DROP COLUMN

PostgreSQL DROP COLUMN: ALTER TABLE কমান্ডটি SQL এ ব্যবহৃত হয় একটি টেবিলের গঠন পরিবর্তন করতে। এর মাধ্যমে আপনি একটি টেবিল থেকে একটি কলাম চিরস্থায়ীভাবে মুছতে পারেন।

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

এখানে:

- table_name: আপনার টেবিলের নাম
- column_name: সেই কলামের নাম যা আপনি মুছতে চান

উদাহরণ:

ধরা যাক, আপনার একটি টেবিল আছে যার নাম students এবং সেখানে একটি কলাম আছে যার নাম address। আপনি এই address কলামটি মুছে ফেলতে চান।

এখন, SQL কমান্ডটি হবে:

```
ALTER TABLE students
DROP COLUMN address;
```

এই কমান্ডটি students টেবিল থেকে address কলামটি মুছে ফেলবে।

গুরুত্বপূর্ণ কিছু পয়েন্ট:

1. ডেটা হারানো: কলামটি মুছে ফেলার পর, ওই কলামের সমস্ত ডেটা স্থায়ীভাবে মুছে যাবে। সুতরাং, কলামটি মুছে ফেলার আগে ডেটা ব্যাকআপ নেওয়া ভাল।
2. ডিপেনডেন্সি: যদি ওই কলামের ওপর অন্য কোন ডেটাবেস অবজেক্ট (যেমন, কনস্ট্রাইন্ট বা ভিউ) নির্ভরশীল থাকে, তাহলে কলামটি মুছে ফেলার সময় ত্রুটি (error) দেখা দিতে পারে।

7. PostgreSQL DELETE

PostgreSQL DELETE: DELETE FROM কমান্ডটি SQL-এ ব্যবহৃত হয় একটি টেবিল থেকে row বা সমস্ত row মুছে ফেলতে। আপনি WHERE শর্ত ব্যবহার করে নির্দিষ্ট rowগুলো মুছে ফেলতে পারেন। Delete value using condition

```
DELETE FROM table_name
WHERE condition;
```

এখনে:

- table_name: সেই টেবিলের নাম থেকে আপনি (row) রেকর্ড মুছতে চান।
- condition: একটি শর্ত যা নির্ধারণ করে কোন (row) রেকর্ডগুলো মুছতে হবে।

মনে রাখবেন, যদি আপনি WHERE শর্ত না ব্যবহার করেন, তবে টেবিলের সমস্ত রেকর্ড মুছে যাবে।

উদাহরণ 1: একটি নির্দিষ্ট শর্তে রেকর্ড মুছে ফেলা

ধরা যাক, আপনার একটি students টেবিল আছে এবং আপনি age কলামের মান 18 বছরের কম এমন ছাত্রদের ডেটা মুছে ফেলতে চান। তাহলে কমান্ড হবে:

```
DELETE FROM students
WHERE age < 18;
```

এই কমান্ডটি students টেবিল থেকে সমস্ত ছাত্রের রেকর্ড মুছে ফেলবে যাদের বয়স 18 এর কম।

উদাহরণ 2: নির্দিষ্ট রেকর্ড মুছে ফেলা

ধরা যাক, আপনি একটি students টেবিল থেকে student_id = 5 যাদের ছাত্রদের row তথ্য মুছে ফেলতে চান:

```
DELETE FROM students
WHERE student_id = 5;
```

এই কমান্ডটি students টেবিল থেকে student_id 5 এর (row) রেকর্ডটি মুছে ফেলবে।

গুরুত্বপূর্ণ পয়েন্ট:

1. ব্যাকআপ নেওয়া: DELETE কমান্ডটি স্থায়ীভাবে ডেটা মুছে ফেলবে। তাই ডেটা মুছার আগে ব্যাকআপ নেওয়া গুরুত্বপূর্ণ।

2. WHERE শর্ত ছাড়া: যদি WHERE শর্ত না দেওয়া হয়, তাহলে টেবিলের সব রেকর্ড মুছে যাবে, যেমন:

```
DELETE FROM students;
```

এই কমান্ডটি students টেবিল থেকে সমস্ত রেকর্ড মুছে ফেলবে, তবে টেবিলটি থাকবে।

3. ট্রানজেকশন: কিছু ডেটাবেস ব্যবস্থায়, আপনি DELETE কমান্ডের পরে ট্রানজেকশন ব্যবহার করে কমান্ডটি ফেরত নিতে (rollback) পারেন, যদি কোনও ভুল হয়ে থাকে।

TRUNCATE TABLE:

TRUNCATE TABLE কমান্ডটি SQL-এ ব্যবহৃত হয় একটি টেবিলের সমস্ত রেকর্ড মুছে ফেলতে, কিন্তু টেবিলের কাঠামো (schema) এবং তার কনস্ট্রাইন্ট, ইনডেক্স ইত্যাদি অপরিবর্তিত থাকে। এর মানে হল যে আপনি সমস্ত ডেটা মুছতে পারবেন, কিন্তু টেবিলটি এবং তার স্ট্রাকচার থাকবে, যাতে আপনি ভবিষ্যতে নতুন ডেটা ইনসার্ট করতে পারবেন।

Syntax:

```
TRUNCATE TABLE table_name;
```

এখানে:

- table_name: সেই টেবিলের নাম যার সমস্ত ডেটা মুছে ফেলতে চান।

উদাহরণ:

ধরা যাক, আপনার একটি students টেবিল আছে এবং আপনি এর সমস্ত রেকর্ড মুছে ফেলতে চান, তবে টেবিলটি অক্ষুণ্ণ রাখতে চান (কাঠামো পরিবর্তন না করে)। তাহলে আপনি এই কমান্ডটি ব্যবহার করবেন:

```
TRUNCATE TABLE students;
```

এই কমান্ডটি students টেবিল থেকে সমস্ত ডেটা মুছে ফেলবে, কিন্তু টেবিলের কাঠামো (যেমন কলামগুলো) ঠিক থাকবে এবং আপনি আবার টেবিলটিতে ডেটা ইনসার্ট করতে পারবেন।

TRUNCATE TABLE এর বৈশিষ্ট্য:

1. ডেটা মুছে ফেলা: এটি টেবিলের সমস্ত রেকর্ড মুছে ফেলবে, কিন্তু টেবিলের কাঠামো পরিবর্তন করবে না।
2. ফাস্ট এবং কার্যকরী: TRUNCATE অনেক দ্রুত এবং দক্ষ হতে পারে কারণ এটি ডেটার প্রতি রেকর্ড অপসারণের পরিবর্তে একটি তাৎক্ষণিক পদ্ধতিতে সমস্ত ডেটা মুছে ফেলে।
3. ডেটা রিকভার করা সম্ভব নয়: একবার TRUNCATE কমান্ড কার্যকর হলে, ডেটা পুনরুদ্ধার করা সম্ভব হয় না (ব্যাকআপ না নেওয়া থাকলে)।
4. ডিলিটের তুলনায় আলাদা: DELETE কমান্ডের সঙ্গে তুলনা করলে, TRUNCATE কমান্ডটি ডেটা মুছে ফেলার ক্ষেত্রে দ্রুত, কিন্তু DELETE মেথড ব্যবহার করলে আপনি WHERE শর্ত দিয়ে নির্দিষ্ট রেকর্ড মুছতে পারেন, যা TRUNCATE-এ সম্ভব নয়।
5. লকিং: TRUNCATE সাধারণত টেবিলের উপর একটি পেজ লক বা টেবিল লক প্রয়োগ করে, যেখানে DELETE প্রতিটি রেকর্ডের জন্য লক ব্যবহার করে।

TRUNCATE vs DELETE:

- DELETE: এটি সিঙ্গেল row মুছে ফেলতে বা শর্ত অনুযায়ী রেকর্ড মুছে ফেলতে সক্ষম। তবে এটি একে একে প্রতিটি row মুছে ফেলায় কম গতিতে কাজ করে।
- TRUNCATE: এটি টেবিলের সমস্ত row একযোগে মুছে ফেলে এবং দ্রুত কাজ করে।

8. PostgreSQL DROP TABLE

PostgreSQL DROP TABLE: DROP TABLE কমান্ডটি SQL-এ ব্যবহৃত হয় একটি টেবিল পুরোপুরি মুছে ফেলতে। এটি টেবিলটি এবং তার সব ডেটা, কাঠামো (schema), এবং সম্পর্কিত ইনডেক্স বা কনস্ট্রেইন্ট মুছে দেয়।

```
DROP TABLE table_name;
```

এখানে:

- table_name: আপনি যেই টেবিলটি মুছে ফেলতে চান তার নাম।

উদাহরণ:

ধরা যাক, আপনার একটি টেবিল আছে যার নাম students এবং আপনি এটি মুছে ফেলতে চান। তাহলে কমান্ড হবে:

```
DROP TABLE students;
```

এই কমান্ডটি students টেবিলটি পুরোপুরি মুছে ফেলবে, যার মধ্যে থাকা সমস্ত ডেটা, টেবিলের কাঠামো এবং অন্যান্য সম্পর্কিত অবজেক্ট (যেমন ইনডেক্স, কনস্ট্রেইন্ট) মুছে যাবে।

গুরুত্বপূর্ণ পয়েন্ট:

1. ডেটা হারানো: DROP TABLE কমান্ডটি টেবিলের সমস্ত ডেটা স্থায়ীভাবে মুছে ফেলবে। এটি পুনরুদ্ধারযোগ্য নয়, তাই টেবিলটি মুছে ফেলার আগে ডেটা ব্যাকআপ নেওয়া গুরুত্বপূর্ণ।
2. রিলেশনাল ডেটাবেসে নির্ভরতা: যদি অন্য কোনো টেবিল বা অবজেক্ট (যেমন, ফোরেন কি বা ভিউ) এই টেবিলটির ওপর নির্ভরশীল থাকে, তাহলে DROP TABLE কমান্ডটি ব্যর্থ হতে পারে।
3. পুনরায় তৈরি করা: টেবিল মুছে ফেলা হলে, আপনাকে যদি আবার সেই টেবিলটি তৈরি করতে হয়, তবে আপনাকে আবার CREATE TABLE কমান্ড ব্যবহার করে নতুন টেবিল তৈরি করতে হবে।

PostgreSQL Query Basics:

PostgreSQL Query Basics বলতে PostgreSQL ডেটাবেসে কুইরি লেখার মৌলিক ধারণাগুলি বোঝানো হয়। এর মধ্যে অন্তর্ভুক্ত থাকে PostgreSQL-এর সিনট্যাক্স এবং অপারেটর সম্পর্কে মৌলিক ধারণা।

1. PostgreSQL Syntax

PostgreSQL Syntax:

PostgreSQL-এর সিনট্যাক্স হচ্ছে সেই নিয়মাবলী বা কাঠামো যার মাধ্যমে আপনি PostgreSQL ডেটাবেসে প্রশ্ন বা কুইরি (Query) লেখেন। একে আমরা "SQL সিনট্যাক্স" ও বলতে পারি, কারণ PostgreSQL SQL ভাষার উপর ভিত্তি করে কাজ করে।

SELECT ব্যবহার করতে হয়।

2. PostgreSQL Operators

PostgreSQL Operators:

PostgreSQL-এ অপারেটর হল বিশেষ চিহ্ন বা শব্দ যা ডেটা ফিল্টার বা প্রসেস করার জন্য ব্যবহৃত হয়। এটি সাধারণত দুটি বা তার বেশি মধ্যে সম্পর্ক স্থাপন করে। PostgreSQL-এ বিভিন্ন ধরনের অপারেটর আছে।
পাইথনে এইগুলা operator আমরা ব্যবহার করেছিলাম। এখন আবার রিভাইস দিবো।

গাণিতিক অপারেটর (Arithmetic Operators):

- 1) +: যোগফল
- 2) -: বিয়োগফল
- 3) *: গুণফল
- 4) /: ভাগফল
- 5) %: ভাগশেষ

তুলনামূলক অপারেটর (Comparison Operators):

- 1) =: সমান
- 2) != বা <>: অসমান
- 3) <: ছোট
- 4) >: বড়
- 5) <=: ছোট বা সমান
- 6) >=: বড় বা সমান

লজিক্যাল অপারেটর (Logical Operators):

- AND: দুটি শর্ত সঠিক হলে তবেই ফলস্বরূপ সত্য
- OR: কোন একটি শর্ত সঠিক হলেই ফলস্বরূপ সত্য
- NOT: শর্তের বিপরীত মান

BETWEEN: এটি একটি শর্তে নির্দিষ্ট রেঞ্জ বা সীমার মধ্যে মান পরীক্ষা করতে ব্যবহৃত হয়।

৫. IN: এটি ব্যবহার করা হয় একাধিক মানের মধ্যে কোন একটি মান খুঁজে বের করার জন্য।

৬. LIKE: এটি ব্যবহার করা হয় একটি প্যাটার্ন মেচ করার জন্য (যেমন স্ট্রিং বা টেক্সটের সাথে ম্যাচ করা)।

সারাংশ:

1. PostgreSQL Syntax: SQL কুইরি লেখার কাঠামো এবং নিয়মাবলী।
2. PostgreSQL Operators: কুইরিতে বিভিন্ন অপারেটর ব্যবহৃত হয় যাতে আপনি ডেটা প্রসেস বা ফিল্টার করতে পারেন (যেমন, গাণিতিক অপারেটর, তুলনামূলক অপারেটর, লজিক্যাল অপারেটর ইত্যাদি)।
এগুলোর মূল লক্ষ্য হল ডেটাবেসে ডেটা সিলেক্ট করা, ফিল্টার করা, আপডেট করা বা মুছে ফেলা।

PostgreSQL Queries

Python এ print() এবং SQL এ SELECT এর সম্পর্ক:

Python ভাষায় print() ফাংশন ব্যবহৃত হয় শুধুমাত্র কোন কিছু স্ক্রীনে প্রদর্শন করার জন্য। এটি কোনো মানের সাথে সংশ্লিষ্ট কোনো পরিবর্তন বা অপারেশন করে না, শুধুমাত্র তার মান বা বিষয়বস্তু প্রদর্শন করে। অর্থাৎ, print() ফাংশন শুধু একটি আউটপুট তৈরি করে, কিন্তু এটি কোনো ভ্যারিয়েবল বা মানের অবস্থা পরিবর্তন করে না।

এভাবে SQL ভাষায় SELECT কমান্ডও একইভাবে কাজ করে। SQL-এ SELECT কমান্ড ব্যবহৃত হয় ডাটাবেস থেকে নির্দিষ্ট তথ্য নির্বাচন বা প্রদর্শন করার জন্য। এটি শুধুমাত্র ডাটাবেসের তথ্য প্রদর্শন করে, কিন্তু ডাটাবেসের মধ্যে কোনো পরিবর্তন বা আপডেট করে না। অর্থাৎ, SELECT কেবলমাত্র ডেটা বের করে আনে, কিন্তু ডাটাবেসে কোনো পরিবর্তন ঘটায় না।

1. PostgreSQL SELECT

SELECT কমান্ড ব্যবহৃত হয় ডাটাবেস থেকে ডেটা নির্বাচন করতে।

গঠন:

```
SELECT column1, column2, ... FROM table_name;
```

উদাহরণ:

```
SELECT name, age FROM employees;
```

এই কোডটি employees টেবিল থেকে name এবং age কলাম গুলোর সকল মান দেখাবে।

2. PostgreSQL SELECT DISTINCT

SELECT DISTINCT ব্যবহৃত হয় ডুপ্লিকেট মান সরিয়ে শুধু একক মানগুলি দেখাতে।

গঠন:

```
SELECT DISTINCT column_name FROM table_name;
```

উদাহরণ:

```
SELECT DISTINCT department FROM employees;
```

এটি employees টেবিল থেকে কোনো department এর নাম একের অধিক দেখাবে না। যদি একের অধিক থাকে, সেগুলো সে দেখাবে না। একই মানকে দুইবার দেখায় না।

3. PostgreSQL WHERE

WHERE word ব্যবহৃত হয় একটি নির্দিষ্ট শর্ত প্ররূপের করার জন্য। SQL এর where পাইথনের if সমতুল্য।

গঠন:

```
SELECT column1, column2, ... FROM table_name WHERE condition/relational operator;
```

উদাহরণ:

```
SELECT name, age FROM employees WHERE age > 30;
```

এটি employees টেবিল থেকে এমন সকল কর্মচারীর নাম এবং বয়স দেখাবে করবে যাদের বয়স 30 এর বেশি।

4. PostgreSQL ORDER BY

ORDER BY ব্যবহৃত হয় ডেটা সাজানোর জন্য (অধিকাংশ সময় ASC বা DESC ব্যবহার করে)। SQL এর order আর python এর sort তারা সাজাতে ব্যবহার করা হয়।

গঠন:

```
SELECT column1, column2, ... FROM table_name ORDER BY column_name [ASC|DESC];
```

উদাহরণ:

```
SELECT name, age FROM employees ORDER BY age DESC;
```

এটি employees টেবিল থেকে name এবং age নির্বাচন করবে এবং বয়সের ক্রমে (বড় থেকে ছোট) সাজাবে।

5. PostgreSQL LIMIT

LIMIT ব্যবহৃত হয় একটি নির্দিষ্ট সংখ্যক রেকর্ড সীমাবদ্ধ করতে।

গঠন:

```
SELECT column1, column2, ... FROM table_name LIMIT number;
```

উদাহরণ:

```
SELECT name FROM employees LIMIT 5;
```

এটি employees টেবিল থেকে প্রথম ৫টি কর্মচারীর নাম দেখাবে।

6. PostgreSQL MIN and MAX

MIN এবং MAX ফাংশন ব্যবহৃত হয় সংখ্যাত্মক কলামের সর্বনিম্ন এবং সর্বোচ্চ মান নির্ধারণ করতে।

গঠন:

```
SELECT MIN(column_name), MAX(column_name) FROM table_name;
```

উদাহরণ:

```
SELECT MIN(age), MAX(age) FROM employees;
```

এটি employees টেবিল থেকে সর্বনিম্ন এবং সর্বোচ্চ বয়স বের করবে।

7. PostgreSQL COUNT

COUNT ফাংশন ব্যবহৃত হয় রেকর্ডের (row) সংখ্যা গণনা করতে।

গঠন:

```
SELECT COUNT(column_name) FROM table_name;
```

উদাহরণ:

```
SELECT COUNT(*) FROM employees;
```

এটি employees টেবিলের মোট রেকর্ড (row) সংখ্যা গণনা করবে।

8. PostgreSQL SUM

SUM ফাংশন ব্যবহৃত হয় একটি সংখ্যাত্মক কলামের যোগফল বের করতে।

গঠন:

```
SELECT SUM(column_name) FROM table_name;
```

উদাহরণ:

```
SELECT SUM(salary) FROM employees;
```

এটি employees টেবিল থেকে সকল কর্মচারীর salary column এর যোগফল বের করবে।

9. PostgreSQL AVG

AVG ফাংশন ব্যবহৃত হয় একটি সংখ্যাত্মক কলামের গড় মান বের করতে।

গঠন:

```
SELECT AVG(column_name) FROM table_name;
```

উদাহরণ:

```
SELECT AVG(salary) FROM employees;
```

এটি employees টেবিল থেকে সকল কর্মচারীর salary এর গড় বের করবে।

10. PostgreSQL LIKE

LIKE ব্যবহৃত হয় প্যাটার্ন ম্যাচিং (যেমন wildcard ব্যবহার করে) ফিল্টার করতে।

গঠন:

```
SELECT column1, column2 FROM table_name WHERE column_name LIKE pattern;
```

উদাহরণ:

```
SELECT name FROM employees WHERE name LIKE 'J%';
```

এটি employees টেবিল থেকে এমন সকল কর্মচারীর নাম নির্বাচন করবে যারা 'J' দিয়ে শুরু হয়।

11. PostgreSQL IN

IN ব্যবহৃত হয় নির্দিষ্ট মানের মধ্যে থেকে কোনো একটি মানের সাথে মেলানোর জন্য। in পাইথনে if এবং for এ ব্যবহার হয়।

গঠন:

```
SELECT column1, column2 FROM table_name WHERE column_name IN (value1, value2, ...);
```

উদাহরণ:

```
SELECT name FROM employees WHERE department IN ('HR', 'IT');
```

এটি employees টেবিল থেকে এমন সকল কর্মচারীর নাম নির্বাচন করবে যারা 'HR' অথবা 'IT' বিভাগে কাজ করেন।

12. PostgreSQL BETWEEN

BETWEEN ব্যবহৃত হয় একটি নির্দিষ্ট সীমার মধ্যে মান দেখাতে।

গঠন:

```
SELECT column1, column2 FROM table_name WHERE column_name BETWEEN value1 AND value2;
```

উদাহরণ:

```
SELECT name FROM employees WHERE age BETWEEN 25 AND 35;
```

এটি employees টেবিল থেকে এমন সকল কর্মচারীর নাম দেখাবে, যাদের বয়স ২৫ থেকে ৩৫ এর মধ্যে।

```
SELECT * FROM Products WHERE product_name BETWEEN 'Pavlova' AND
'Tofu' Order by product_name;
```

13. PostgreSQL AS

AS ব্যবহৃত হয় একটি কলাম বা টেবিলের জন্য অ্যালিয়াস (alternative name) তৈরি করতে। alias এর মাধ্যমে পাইথনের মতন সংক্ষিপ্ত করা যায়।

গঠন:

```
SELECT column_name AS alias_name FROM table_name;
```

উদাহরণ:

```
SELECT name AS employee_name, age AS employee_age FROM employees;
```

এটি employees টেবিল থেকে name এবং age কলাম গুলি নির্বাচন করবে, কিন্তু তাদের অ্যালিয়াস হবে employee_name এবং employee_age।