SQL পরিচিতি

রিলেশনাল ডেটাবেজ(Relational Database) এ তথ্য জমা(store), পুনরুদ্ধার(retrive) এবং পরিচালনার(manipulating) জন্য **SQL** একটি স্টান্ডার্ড ভাষা।

SQL কি?

SQL হলো স্ট্রাকচার্ড কুয়েরি ল্যাঙ্গুয়েজ(**S**tructured **Q**uery **L**anguage) যা রিলেশনাল ডেটাবেজে সঞ্চিত ডেটা সংরক্ষণ, পুনরুদ্ধার এবং পরিচালনার জন্য ব্যবহৃত একটি স্টান্ডার্ড ভাষা।

MySQL, SQL Server, Access, Oracle, Sybase ইত্যাদি রিলেশনাল ডেটাবেজ ম্যানেজমেন্ট সিস্টেম-সমূহ(RDBMS) SQL কে স্টান্ডার্ড ভাষা হিসাবে ব্যবহার করে।

- SQL এর পূর্ণরূপঃ **S**tructured **Q**uery **L**anguage ।
- SQL এর মাধ্যমে আপনি রিলেশনাল ডেটাবেজে এক্সেস করতে পারবেন।
- SQL একটি ANSI(**A**merican **N**ational **S**tandards **I**nstitute) স্ট্যান্ডার্ড।

SQL কি কি করতে পারে?

- SQL ইউজারকে রিলেশনাল ডেটাবেজ ম্যানেজমেন্ট সিস্টেম থেকে ডেটা এক্সেস করতে অনুমতি দেয়।
- SQL ডেটাবেজে কুয়েরি সম্পাদন করতে পারে।
- SQL নতুন ডেটাবেজ তৈরি করতে পারে।
- SQL ডেটাবেজে নতুন টেবিল তৈরি করতে পারে।
- SQL ডেটাবেজ থেকে তথ্য পুনরূদ্ধার করতে পারে।
- SQL ডেটাবেজে তথ্য সংরক্ষন করতে পারে।
- SQL ডেটাবেজে তথ্য হালনাগাদ করতে পারে।
- SQL ডেটাবেজ থেকে তথ্য মুছে ফেলতে পারে।
- SQL ডেটাবেজের মধ্যে তথ্য সংরক্ষণ পদ্ধতি তৈরি করতে পারে।
- SQL ডেটাবেজের ভিউ(view) তৈরি করতে পারে।
- SQL ডেটাবেজে টেবিল, কার্যপ্রনালী এবং ভিউ এর উপর পারমিশন সেট করতে পারে।
- SQL ডেটাবেজে যে কোন কার্য-সম্পাদন করতে পারে।

SQL একটি স্ট্যান্ডার্ড

SQL ভাষা ANSI(American National Standards Institute) স্ট্যান্ডার্ড হওয়া সত্ত্বেও এর কিছু ভিন্ন ভার্সনও রয়েছে।

যাইহোক, ANSI স্ট্যান্ডার্ড মেনে চলার জন্য SQL এর সকল ভার্সন-ই প্রধান প্রধান কমান্ড-সমূহ যেমন- `CREATE`, `SELECT`, `UPDATE`, `DELETE`, `INSERT`, `WHERE` ইত্যাদি সাপোর্ট করে।

বিঃদ্রঃ অধিকাংশ SQL ডেটাবেজ প্রোগ্রামের SQL স্ট্যান্ডার্ড ছাড়াও তাদের নিজস্ব কিছু এক্সটেনশন রয়েছে।

ওয়েব সাইটে SQL এর ব্যবহার

ডেটাবেজ থেকে তথ্য দেখাবে এমন একটি ওয়েব-সাইট তৈরী করতে যা প্রয়োজন হবেঃ

- একটি RDBMS ডেটাবেজ প্রোগ্রাম। যেমন- MS Access, SQL Server, MySQL ইত্যাদি।
- একটি সার্ভার সাইড স্ক্রিপ্টিং ভাষা। যেমন- PHP অথবা ASP
- ডেটাবেজ থেকে যে কোনো তথ্য পেতে আপনাকে SQL(Sql) ব্যবহার করতে হবে।
- এছাড়া ডায়নামিকভাবে তথ্য এক্সেস করতে চাইলে SQL এর সাথে Ajax অথবা Jquery-ও ব্যবহার করতে পারেন।

RDBMS

RDBMS এর পূর্ণরূপঃ **R**elational **D**atabase **M**anagement **S**ystem.

RDBMS হলো SQL এর ভিত্তি এবং সকল মর্ডান ডেটাবেজ সিস্টেমেরও ভিত্তি। যেমন- MS SQL Server, IBM DB2, Oracle, MySQL এবং Microsoft Access।

তথ্য-সমূহ RDBMS ডেটাবেজ এর অবজেক্টে সংরক্ষিত থাকে, আমাদের কাছে এই অবজেক্টগুলো টেবিল নামে পরিচিত। একটি টেবিল সম্মন্ধযুক্ত কিছু তথ্যের(data) সংগ্রহ যা কলাম(field) এবং সারি(tuple/record) নিয়ে গঠিত। আমাদের সকল তথ্য ডেটাবেজের এই কলাম এবং সারির মধ্যেই সংরক্ষিত হয়/থাকে।

SQL গঠনপ্রণালী

ডেটাবেজ টেবিল

একটি ডেটাবেজে প্রায়ই এক বা একাধিক টেবিল থাকে। প্রতিটি টেবিলকেই নির্দিষ্ট নাম দ্বারা শনাক্ত করা হয়। যেমন- "Student_details" অথবা "Student_result"। টেবিলের প্রতিটি সারি তথ্য নিয়ে গঠিত হয়।

উদাহরণ হিসাবে আমরা আমাদের টিউটোরিয়ালে একটি নমুনা ডেটাবেজ ব্যবহার করবো।

উদাহরণস্বরুপঃ নিচের অংশটুকু নমুনা ডেটাবেজের "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

ক্রমিক নং	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

উপরের টেবিলটিতে পাঁচটি সারি(প্রতি সারিতে একজন শিক্ষার্থী) এবং চারটি কলাম রয়েছে (ক্রমিক নং, শিক্ষার্থীর নাম, প্রতিষ্ঠানের নাম এবং ঠিকানা) যাহাতে ৫ জন শিক্ষার্থীর সম্পর্কে বিস্তারিত তথ্য রয়েছে।

SQL স্টেটমেন্ট(কমান্ড)

SQL স্টেটমেন্ট(statement) এর মাধ্যমে ডেটাবেজের বেশির ভাগ ক্রিয়াকলাপ সম্পাদিত হয়।

নিচের SQL স্টেটমেন্টটি "Student_details" টেবিল থেকে সকল রেকর্ড সিলেক্ট করবেঃ

উদাহরণ

```
SELECT * FROM Student_details;
Copy
```

আমাদের এই টিউটোরিয়ালে অধিকাংশ SQL স্টেটমেন্ট পর্যায়ক্রমে তুলে ধরা হয়েছে।

SQL কীওয়ার্ড-সমূহ কেস-সেন্সিটিভ নয়। যেমনঃ `select` এবং `SELECT` একই অর্থে ব্যবহৃত হয়।

আমাদের এই টিউটোরিয়ালে আমরা SQL কীওয়ার্ড-সমূহকে বড়-হাতের অক্ষরে লিখবো। এর দরুন আমরা SQL কীওয়ার্ড-সমূহকে SQL স্টেটমেন্টের মধ্যে খুব সহজেই শনাক্ত করতে পারবো।

সেমিকোলন(;) এর ব্যবহার

কিছু কিছু ডেটাবেজ সিস্টেমে প্রতিটি SQL স্টেটমেন্টের শেষে একটি **সেমিকোলন(;)** ব্যবহার করতে হয়।

ডেটাবেজ সিস্টেমে স্টেটমেন্ট-সমূহকে আলাদা করার আদর্শ উপায় হলো **সেমিকোলন(;)** যা ডেটাবেজ সিস্টেম সার্ভারে একই সাথে একাধিক SQL স্টেটমেন্ট এক্সিকিউট করতে পারে।

এই টিউটোরিয়ালের প্রতিটি SQL স্টেটমেন্টের শেষে আমরা সেমিকোলন ব্যবহার করবো।

সচরাচর ব্যবহৃত SQL কমান্ড-সমূহ

- **USE** - ডিফল্ট ডেটাবেজ সিলেক্ট করে।
- **DESCRIBE** - ডেটাবেজের টেবিলের গঠন দেখায়।
- **SELECT** - ডেটাবেজ থেকে তথ্য পুনরুদ্ধার(retrieve) করে।
- **UPDATE** - ডেটাবেজের তথ্য আপডেট করে।
- **DELETE** - ডেটাবেজ থেকে তথ্য ডিলেট করে।
- **INSERT INTO** - ডেটাবেজে নতুন তথ্য প্রবেশ করায়।
- **CREATE DATABASE** - নতুন ডেটাবেজ তৈরি করে।
- **ALTER DATABASE** - ডেটাবেজ পরিবর্তন করে।
- **CREATE TABLE** - নতুন টেবিল তৈরি করে।
- **ALTER TABLE** - টেবিল পরিবর্তন করে।
- **DROP TABLE** - টেবিল ডিলেট করে।
- **CREATE INDEX** - ইন্ডেক্স তৈরি করে।
- **DROP INDEX** - ইন্ডেক্স ডিলেট করে।

SQL CREATE DATABASE স্টেটমেন্ট

একটি ডেটাবেজ তৈরি করতে `CREATE DATABASE` স্টেটমেন্ট ব্যবহার করা হয়।

SQL CREATE DATABASE সিনট্যাক্স

```
CREATE DATABASE name_of_database;
Copy
```

SQL CREATE DATABASE উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "student" নামে একটি ডেটাবেজ তৈরি করেঃ

```
CREATE DATABASE student;
Copy
```

`CREATE TABLE` স্টেটমেন্টটি ডেটাবেজে নতুন টেবিল তৈরি করে।

SQL CREATE TABLE স্টেটমেন্ট

ডেটাবেজে নতুন টেবিল তৈরি করতে `CREATE TABLE` স্টেটমেন্ট ব্যবহার করা হয়।

একটি টেবিল সাধারণত কলাম এবং সারি নিয়ে গঠিত হয় এবং সনাক্ত করার জন্য প্রতিটি টেবিলের অবশ্যই একটি নাম থাকতে হবে।

SQL CREATE TABLE সিনটেক্স

```
CREATE TABLE name_of_table(
        name_of_column_1 data_type(size),
    name_of_column_2 data_type(size),
    name_of_column_3 data_type(size),
....
);
Copy
```

"name_of_column" প্যারামিটারটি কলামের নাম ঠিক করে। "data_type" প্যারামিটারটি কলামের ডেটার টাইপ ঠিক করে। যেমন- `varchar`, `integer`, `decimal`, `date`, `text` ইত্যাদি। "size" প্যারামিটারটি কলামের দৈর্ঘ্য ঠিক করে অর্থাৎ কলামের ডেটা কতটি অক্ষর ধারন করবে তা নির্ধারন করে।

SQL CREATE TABLE উদাহরণ

এখন আমরা "Student_details" নামে একটি টেবিল তৈরি করবো যার মধ্যে ৫টি কলাম থাকবেঃ "id", "roll_number", "student_name", "institute" এবং "address"।

আমরা নিম্নে `CREATE TABLE` স্টেটমেন্টটি ব্যবহার করবোঃ

উদাহরণ

```
CREATE TABLE Student_details(
  id int auto_increment,
  roll_number varchar(255),
  student_name varchar(255),
  institute varchar(255),
  address varchar(255)
);
Copy
```

"id" কলামের ডেটা টাইপ `int` হওয়ায় এটি শুধুমাত্র পূর্ণসংখ্যা জমা রাখবে। "id" এর `auto_increment` নিয়ে আমরা পরবর্তীতে আলোচনা করবো। "roll_number", "student_name", "institute" এবং "address" কলাম গুলোর ডেটা টাইপ `varchar` হওয়ায় এরা অক্ষর/বর্ণ জমা রাখবে এবং এদের সর্বোচ্চ দৈর্ঘ্য হবে ২৫৫টি বর্ণ।

ফাঁকা "Student_details" টেবিলটি নিম্নের ন্যায় দেখাবেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা

বিঃদ্রঃ ফাঁকা টেবিলটিতে এখন `INSERT INTO` স্টেটমেন্টের মাধ্যমে তথ্য পূর্ণ করা যাবে।

SQL DROP স্টেটমেন্ট

`DROP` স্টেটমেন্টটি ব্যবহার করে খুব সহজেই ইনডেক্স, টেবিল এবং ডেটাবেজ ডিলেট করা যায়।

DROP INDEX স্টেটমেন্ট

`DROP INDEX` স্টেটমেন্টটি ব্যবহার করে একটি টেবিলের ইনডেক্স ডিলেট করা হয়।

MS Access এর জন্য DROP INDEX সিনট্যাক্স

```
DROP INDEX name_of_index ON name_of_table
Copy
```

SQL Server এর জন্য DROP INDEX সিনট্যাক্সঃ

```
DROP INDEX name_of_table.name_of_index
Copy
```

DB2/Oracle এর জন্য DROP INDEX সিনট্যাক্সঃ

```
DROP INDEX name_of_index
Copy
```

MySQL এর জন্য DROP INDEX সিনট্যাক্সঃ

```
ALTER TABLE name_of_table DROP INDEX name_of_index
Copy
```

DROP TABLE স্টেটমেন্ট

`DROP TABLE` স্টেটমেন্টটি ব্যবহার করে একটি টেবিল ডিলেট করা হয়।

```
DROP TABLE name_of_table
Copy
```

DROP DATABASE স্টেটমেন্ট

`DROP DATABASE` স্টেটমেন্টটি ব্যবহার করে একটি ডেটাবেজ ডিলেট করা হয়।

```
DROP DATABASE name_of_database
Copy
```

TRUNCATE TABLE স্টেটমেন্ট

যদি আপনার শুধুমাত্র টেবিলের তথ্য-সমূহ ডিলেট করার প্রয়োজন হয় তাহলে আপনি কী করবেন?

সেক্ষেত্রে আপনি `TRUNCATE TABLE` স্টেটমেন্টটি ব্যবহার করতে পারেনঃ

```
TRUNCATE TABLE name_of_table
```

SQL SELECT স্টেটমেন্ট

ডেটাবেজ থেকে তথ্য(Data) সিলেক্ট/পুনরূদ্ধার করতে SQL `SELECT` স্টেটমেন্টটি ব্যবহার করা হয়।

SQL SELECT স্টেটমেন্ট

`SELECT` স্টেটমেন্টটি ডেটাবেজ থেকে তথ্য সিলেক্ট করে।

ডেটাবেজ থেকে রিটার্ন ডেটা ফলাফল টেবিলে জমা হয়, যাহাকে আমরা `result-set` বলে থাকি।

SQL SELECT স্টেটমেন্টের সিনট্যাক্স

```
SELECT name_of_column, name_of_column FROM name_of_table;
Copy
```

এবং

```
SELECT * FROM name_of_table;
Copy
```

এখানে **name_of_column** হলো টেবিলের মধ্যে কলামের নাম এবং **name_of_table** হলো টেবিলের নাম।

নমুনা ডেটাবেজ

`SELECT` স্টেটমেন্টের ব্যবহার দেখার জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

উদাহরণস্বরূপঃ `SELECT` স্টেটমেন্ট ব্যবহার করে "Student_details" টেবিল থেকে নিচের অংশটুকু নেওয়া(select) হয়েছে।

ক্রমিক নং	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

নির্দিষ্ট কলাম SELECT করা

নিম্নের SQL `SELECT` স্টেটমেন্টটি "Student_details" টেবিলের "শিক্ষার্থীর নাম(Student_name)" এবং "প্রতিষ্ঠানের নাম(Institute)" এই দুই কলামকে সিলেক্ট করবেঃ

উদাহরণ

```
SELECT Student_name, Institute FROM Student_details;
Copy
```

সকল(*) কলাম SELECT করা

নিম্নের `SELECT` স্টেটমেন্টটি "Student_details" টেবিলের সকল কলামকে সিলেক্ট করবেঃ

উদাহরণ

```
SELECT * FROM Student_details;
Copy
```

Result-set ন্যাভিগেশন

ডেটাবেজ সফটওয়ার সিস্টেমে `Result-set` ন্যাভিগেশন ব্যবহার করা হয়। এটি প্রোগ্রামিং ফাংশন। এর গুরুত্বপূর্ণ কিছু ফাংশন হলোঃ `Move-To-First-Record`, `Get-Record-Content`, `Move-To-Next-Record`, `Get-Record-Count` ইত্যাদি।

SQL INSERT INTO স্টেটমেন্ট

`INSERT INTO` স্টেটমেন্টটি ব্যবহার করে ডেটাবেজের টেবিলে নতুন তথ্য সংযোগ(insert) করা যায়।

SQL INSERT INTO স্টেটমেন্ট

`INSERT INTO` স্টেটমেন্টটি ব্যবহার করে ডেটাবেজের টেবিলে নতুন তথ্য সংযোগ করা যায়।

INSERT INTO সিনট্যাক্স

`INSERT INTO` স্টেটমেন্টটি দুইভাবে লেখা যেতে পারে।

পদ্ধতি১ঃ নিম্নের সিনট্যাক্সে শুধুমাত্র ভ্যালু নির্দিষ্ট করে দেওয়া হয়েছে। কলামের নাম নির্দিষ্ট করে দেওয়া হয় নি।

```
INSERT INTO name_of_table VALUES (value1,value2,value3,...);
Copy
```

পদ্ধতি২ঃ নিম্নের সিনট্যাক্সে কলাম নাম এবং এর ভ্যালু নির্দিষ্ট করে দেওয়া হয়েছেঃ

```
INSERT INTO name_of_table (name_of_column1,name_of_column2,name_of_column3,...)
VALUES (value1,value2,value3,...);
Copy
```

বিঃদ্রঃ টেবিলে ডেটা `INSERT` করার সময় যদি কলামের নাম ব্যবহার না করেন তাহলে ভ্যালু-সমূহের ক্রম কলাম অনুযায়ী ঠিক রাখতে হবে।

নমুনা ডাটাবেস

`INSERT INTO` স্টেটমেন্টের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ Student ব্যবহার করবো।

উদাহরণস্বরূপঃ `SELECT` স্টেটমেন্ট ব্যবহার করে "Student_details" টেবিল থেকে নিচের অংশটুকু নেওয়া(select) হয়েছে।

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

টেবিলে নতুন তথ্য INSERT করা

নিম্নের কোড ব্যবহার করে "Student_details" টেবিলে আপনি একটি নতুন সারি(রেকর্ড) যোগ করতে পারবেন।

উদাহরণ

```
INSERT INTO Student_details (Roll_number, Student_name, Institute, Address)
VALUES ('১০৬','নাসির হোসেন','জাতীয় বিশ্ববিদ্যালয়','চাঁদপুর');
Copy
```

এখন "Student_details" টেবিলের তথ্যগুলো এমন দেখাবেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা

১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৬	১০৬	নাসির হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

আপনি কি লক্ষ্য করেছেন যে, "আইডি নং(Id)" ফিল্ডে আমরা কোনো নম্বর ইনপুট দেইনি?
টেবিলের "আইডি নং(Id)" কলামটিতে `AUTO_INCREMENT` সেট করার ফলে টেবিলে নতুন রেকর্ড যোগ হলেই এর ভ্যালু স্বয়ংক্রিয়ভাবে এক বৃদ্ধি পাবে।

নির্দিষ্ট কলামে তথ্য ইনপুট করা

শুধুমাত্র নির্দিষ্ট কিছু কলামেও তথ্য ইনপুট করা সম্ভব।

নিম্নের `INSERT INTO` স্টেটমেন্টটি "**Student_details**" টেবিলে একটি নতুন রেকর্ড যুক্ত করবে। কিন্তু শুধুমাত্র "শিক্ষার্থীর নাম(Student_name)", "প্রতিষ্ঠানের নাম(Institute)" এবং "ঠিকানা(Address)" কলামে তথ্য ইনপুট নিবে এবং "Id" কলামটি স্বয়ংক্রিয়ভাবে এর ভ্যালু এক বদ্ধি করে নিজেকে আপডেট করে নিবেঃ

উদাহরণ

```
INSERT INTO Student_details (Student_name, Institute)
VALUES ('মোঃ ফয়সাল ইসলাম','জাতীয় বিশ্ববিদ্যালয়', 'রাজশাহী');
Copy
```

এখন "Student_details" টেবিলের তথ্য গুলো এমন দেখাবেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৬	১০৬	নাসির হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৭	খালি(null)	মোঃ ফয়সাল ইসলাম	জাতীয় বিশ্ববিদ্যালয়	রাজশাহী

SQl WHERE Clause

রেকর্ড ফিল্টার করার জন্য SQL `WHERE clause` ব্যবহৃত হয়।

SQl WHERE Clause

নির্দিষ্ট শর্ত সাপেক্ষে ডেটাবেজ থেকে তথ্য পুনরুদ্ধারের জন্য SQL `WHERE clause` ব্যবহার করা হয়। কেবল শর্ত পূরন হলেই আপনি আপনার কাংখিত ফলাফল পাবেন।

SQl WHERE সিনট্যাক্স

```
SELECT name_of_column, name_of_column
FROM name_of_table
WHERE name_of_column operator value;
Copy
```

`WHERE clause` শুধু তথ্য সিলেক্টের জন্য নয় , বরং তথ্য আপডেট এবং ডিলেট করার জন্যও ব্যবহার করা হয়।

বিঃদ্রঃ SQL এর `WHERE clause` অন্যান্য প্রোগ্রামিং ল্যাঙ্গুয়েজ এর কন্ডিশনাল(If) স্টেটমেন্টের মতই।

নমুনা ডেটাবেজ

`WHERE clause` এর ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ Student ব্যবহার করবো।

WHERE Clause এর মাধ্যমে তথ্য সিলেক্ট করা

নিম্নের SQl `WHERE` স্টেটমেন্টটি "Student_details" টেবিলের "ঢাকা" ঠিকানা অন্তর্ভুক্ত সকল শিক্ষার্থীকে সিলেক্ট করবেঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE Address="ঢাকা";
Copy
```

টেক্সট বনাম সংখ্যা

SQl এ টেক্সট লেখার জন্য একক উদ্ধৃতির প্রয়োজন হয়। অনেক ডেটাবেজে ডাবল উদ্ধৃতি চিহ্নের প্রয়োজনও হতে পারে।

যাইহোক, সাংখ্যার ক্ষেত্রে উদ্ধৃতি চিহ্ন(' ') ব্যবহারের প্রয়োজন হয় না।

উদাহরণ

```
SELECT * FROM Student_details
WHERE Id=1;
Copy
```

WHERE Clause অপারেটর

নিম্নের অপারেটর গুলো WHERE clause এ ব্যবহার করা হয়ঃ

অপারেটর	বর্ণনা
=	সমান
<>	সমান না। **বিঃদ্রঃ** কিছু SQl ভার্সনে এই অপারেটকে != হিসেবে ব্যবহার করা হয়
>	বড়
<	ছোট
>=	বড় অথবা সমান
<=	ছোট অথবা সমান
BETWEEN	একটি নির্দিষ্ট সীমার মধ্যে
LIKE	সার্চ এর জন্য প্যাটার্ন
IN	একটি কলামের জন্য একাধিক সম্ভাব্য মান উল্লেখ করা

SQL AND এবং OR অপারেটর

ডেটাবেজের তথ্য গুলোকে এক বা একাধিক শর্ত দ্বারা ফিল্টার করার জন্য AND এবং OR অপারেটর দুটি ব্যবহার করা হয়।

SQL AND এবং OR অপারেটর

AND অপারেটরটি তখনই তথ্য গুলো দেখাবে যখন এর প্রথম এবং দ্বিতীয় শর্তটি সত্য হবে।

OR অপারেটরটি তখনই তথ্য গুলো দেখাবে যখন এর প্রথম অথবা দ্বিতীয় শর্তের মধ্যে যেকোন একটি শর্ত সত্য হবে।

নমুনা ডেটাবেজ

AND এবং OR অপারেটর এর ব্যবহার দেখার জন্য আমরা আমাদের নমুনা ডেটাবেজ Student ব্যবহার করবো।

AND অপারেটরের উদাহরণ

নিচের SQL AND স্টেটমেন্টি ডেটাবেজের "student_details" টেবিল থেকে প্রতিষ্ঠান "জাতীয় বিশ্ববিদ্যালয়" এবং ঠিকানা "ঢাকা" এর অন্তর্ভুক্ত তথ্য গুলোকে সিলেক্ট করবেঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE institute="জাতীয় বিশ্ববিদ্যালয়"
AND Address="ঢাকা";
Copy
```

OR অপারেটরের উদাহরণ

নিচের SQL OR স্টেটমেন্টি ডেটাবেজের "Student_details" টেবিল থেকে "চাঁদপুর" অথবা "ঢাকা" ঠিকানার অন্তর্ভুক্ত তথ্য গুলোকে সিলেক্ট করবেঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE Address="চাঁদপুর"
OR Address="ঢাকা";
Copy
```

AND এবং OR একত্রে ব্যবহার

আপনি AND এবং OR অপারেটর দুটিকে একত্রে ব্যবহার করতে পারেন। যদি অধিক জটিল এক্সপ্রেশন হয় সে ক্ষেত্রে প্রথম বন্ধনী ব্যবহার করতে পারেন।

নিচের SQL স্টেটমেন্টি ডেটাবেজের "Student_details" টেবিল থেকে প্রতিষ্ঠান "জাতীয় বিশ্ববিদ্যালয" এবং "চাঁদপুর" অথবা "ঢাকা" ঠিকানার অন্তর্ভুক্ত তথ্য গুলোকে সিলেক্ট করবেঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE institute="জাতীয় বিশ্ববিদ্যালয়"
AND (Address="চাঁদপুর" OR Address="ঢাকা");
```

SQL UPDATE স্টেটমেন্ট

`UPDATE` স্টেটমেন্টটি ব্যবহার করে টেবিলের তথ্য গুলোকে আপডেট করা যায়।

UPDATE স্টেটমেন্ট

ডেটাবেজের তথ্য আপডেট করার জন্য `UPDATE` স্টেটমেন্ট ব্যবহার করা হয়।

SQL UPDATE সিনট্যাক্স

```
UPDATE name_of_table
SET name_of_column1=value1, name_of_column2=value2, ...
WHERE column=value;
Copy
```

লক্ষ্য করলে দেখবেন যে, `UPDATE` স্টেটমেন্ট এর মধ্যে `WHERE` কন্ডিশনটি ব্যবহার করা হয়েছে!
কোন কোন তথ্যগুলো আপডেট করতে হবে নির্দিষ্ট করে দেওয়ার জন্য `WHERE` কন্ডিশনটি ব্যবহার করা হয়। যদি এটি ব্যবহার করা না হয় তাহলে ডেটাবেজে অবস্থিত সকল তথ্য আপডেট হয়ে যাবে!

নমুনা ডেটাবেজ

`UPDATE` স্টেটমেন্টের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ Student ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

UPDATE উদাহরণ

নিচের `UPDATE` স্টেটমেন্টের মাধ্যমে "Student_details" টেবিলের "ফরহাদ উদ্দিন" এর "Roll_number" এবং "Address" কলামের তথ্য আপডেট করা হয়েছে।

উদাহরণ

```
UPDATE Student_details
SET Roll_number="১৩১", Address="চাঁদপুর"
WHERE Student_name="ফরহাদ
উদ্দিন";
Copy
```

এখন "Student_details" টেবিলের তথ্য গুলো এমন দেখাবেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১৩১	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

Update সতর্কতা!

ডেটাবেজের তথ্য আপডেট করার সময় অবশ্যই সতর্কতা অবলম্বন করা উচিত। তথ্য আপডেট করার সময় যদি `WHERE` কন্ডিশনটি ব্যবহার না করেন তাহলে সমস্ত রেকর্ড আপডেট হয়ে যাবে!

উদাহরণ

```
UPDATE Student_details
SET Roll_number="১৩১", Address="ঢাকা";
Copy
```

এখন "Student_details" টেবিলের তথ্য গুলো এমন দেখাবেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১৩১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	ঢাকা
২	১৩১	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	ঢাকা
৩	১৩১	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	ঢাকা

৪	১৩১	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	ঢাকা
৫	১৩১	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	ঢাকা

SQL DELETE স্টেটমেন্ট

`DELETE` স্টেটমেন্ট ব্যবহার করে টেবিলের তথ্য ডিলেট করা যায়।

SQL DELETE স্টেটমেন্ট

`DELETE` স্টেটমেন্টটি ব্যবহার করে টেবিলের সারি গুলোকে ডিলেট করা যায়।

SQL DELETE সিনট্যাক্স

```
DELETE FROM name_of_table
WHERE column=value;
Copy
```

লক্ষ্য করলে দেখবেন যে, `DELETE` স্টেটমেন্ট এর মধ্যে `WHERE` কন্ডিশনটি ব্যবহার করা হয়েছে!
কোন কোন তথ্যগুলো আপডেট করতে হবে নির্দিষ্ট করে দেওয়ার জন্য `WHERE` কন্ডিশনটি ব্যবহার করা হয়। যদি এটি ব্যবহার করা না হয় তাহলে ডেটাবেজে অবস্থিত সকল তথ্য ডিলেট হয়ে যাবে!

নমুনা ডেটাবেজ

`DELETE` স্টেটমেন্টের ব্যবহার দেখার জন্য আমরা আমাদের নমুনা ডেটাবেজ Student ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

DELETE উদাহরণ

নিচের DELETE স্টেটমেন্টের মাধ্যমে "Student_details" টেবিলের "তামজীদ হাসান" এর তথ্য delete করা হয়েছে।

উদাহরণ

```
DELETE FROM Student_details
WHERE Student_name="তামজীদ হাসান";
Copy
```

এখন "Student_details" টেবিলের তথ্য গুলো এমন দেখাবেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	ঢাকা
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	ঢাকা
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	ঢাকা
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	ঢাকা

সকল তথ্য ডিলেট করা

একটি টেবিলকে ডিলেট না করেই এর সকল তথ্য ডিলেট করা সম্ভব। অর্থাৎ টেবিলের গঠন, এট্রিবিউট এবং ইন্ডেক্স গুলো যথাযথ অবস্থায় থেকে যাবে কিন্তু এর মধ্যে অন্তর্ভুক্ত তথ্য গুলো ডিলেট হয়ে যাবেঃ

```
DELETE FROM name_of_table;
Copy
```
অথবা

```
DELETE * FROM name_of_table;
Copy
```

বিঃদ্রঃ তথ্য ডিলেট করার সময় অবশ্যই সর্তকতা অবলম্বন করতে হবে, কারন একবার তথ্য ডিলেট হয়ে গেলে তা আর পুনরায় ফিরানো সম্ভব না!

SQL LIKE অপারেটর

একটি কলাম থেকে নির্দিষ্ট প্যাটার্ন অনুযায়ী তথ্য অনুসন্ধান করার জন্য WHERE clause এর সাথে SQL LIKE অপারেটরটি ব্যবহার করা হয়।

SQL LIKE অপারেটর

SQL LIKE অপারেটরটি একটি কলাম থেকে নির্দিষ্ট প্যাটার্ন অনুযায়ী তথ্য অনুসন্ধান করে।

SQL LIKE সিনট্যাক্স

```
SELECT name_of_column's
FROM name_of_table
WHERE name_of_column LIKE pattern;
Copy
```

নমুনা ডেটাবেজ

LIKE অপারেটর এর ব্যবহার দেখার জন্য আমরা আমাদের নমুনা ডেটাবেজ Student ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

SQL LIKE অপারেটরের উদাহরণ

নিম্নের SQL LIKE স্টেটমেন্টটি "Student_details" টেবিল থেকে "শিক্ষার্থীর নাম(Student_name)" কলামে অবস্থিত প্রথম অক্ষর "ম" বিশিষ্ট সকল তথ্যকে নিয়ে আসবেঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE Student_name LIKE 'ম%';
Copy
```

বিঃদ্রঃ "%" চিহ্নটি প্যাটার্ন(pattern) এর পূর্বে অথবা পরে ওয়াইল্ডকার্ড(wildcard) নির্ধারণ করতে ব্যবহৃত হয়। পরবর্তীতে আপনি ওয়াইল্ডকার্ড সম্পর্কে আরো বেশী জানবেন।

নিম্নের SQL `LIKE` স্টেটমেন্টটি "Student_details" টেবিল থেকে "শিক্ষার্থীর নাম(Student_name)" কলামে অবস্থিত শেষ অক্ষর "ন" বিশিষ্ট সকল তথ্যকে নিয়ে আসবেঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE Student_name LIKE '%ন';
Copy
```

নিম্নের SQL `LIKE` স্টেটমেন্টটি "Student_details" টেবিল থেকে "ঠিকানা(Address)" কলামে অবস্থিত যে সকল তথ্য গুলোতে "ঢাকা" রয়েছে ঐ সকল তথ্যকে নিয়ে আসবেঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE Address LIKE '%ঢাকা%';
Copy
```

NOT কীওয়ার্ড

`NOT` কিওয়ার্ড দ্বারা আপনি ঐ সকল তথ্য নিয়ে আসতে পারবেন যা প্যাটার্নের সাথে মিলে না।

নিম্নের SQL `LIKE` স্টেটমেন্টটি "Student_details" টেবিল থেকে "ঠিকানা(Address)" কলামে অবস্থিত যে সকল তথ্য গুলোতে "চাঁদপুর" অবস্থিত ঐ সকল তথ্যকে নিবেনাঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE Address NOT LIKE '%চাঁদপুর%';
```

SQL SELECT TOP স্টেটমেন্ট

SQL `SELECT TOP` কমান্ডটি ব্যবহার করে ডেটাবেজ থেকে নির্দিষ্ট সংখ্যক তথ্য পাওয়া যায়। হাজার হাজার তথ্য সম্বলিত টেবিল থেকে কাংখিত তথ্য খুঁজে বের করার জন্য `SELECT TOP` কমান্ডটি ব্যবহার করা হয়।

বিঃদ্রঃ সকল ডেটাবেজে `SELECT TOP` কমান্ডটি সার্পোট করে না। MySQL এর পরিবর্তে `LIMIT` ব্যবহার করে এবং Oracle এর পরিবর্তে `ROWNUM` ব্যবহার করে।

MS Access/SQL Server সিনট্যাক্স

```
SELECT TOP number name_of_column's
FROM name_of_table;
Copy
```

অথবা

```
SELECT TOP percent name_of_column's
FROM name_of_table;
Copy
```

MySQL সিনট্যাক্স

```
SELECT name_of_column's
FROM name_of_table
LIMIT number;
Copy
```

উদাহরণ

```
SELECT *
FROM Student_details
LIMIT 5;
Copy
```

Oracle সিনট্যাক্স

```
SELECT name_of_column's
FROM name_of_table
WHERE ROWNUM <= number;
Copy
```

উদাহরণ

```
SELECT *
FROM Student_details
WHERE ROWNUM <=5;
Copy
```

নমুনা ডেটাবেজ

`SELECT TOP` স্টেটমেন্টের ব্যবহার দেখার জন্য আমরা আমাদের নমুনা ডেটাবেজ Student ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

SQL SELECT TOP উদাহরণ

নিম্নের SQL `SELECT TOP` স্টেটমেন্টটি "Student_details" টেবিল থেকে প্রথম দুইটি রেকর্ডকে সিলেক্ট করবেঃ

উদাহরণ

```
SELECT TOP 2 * FROM Student_details;
Copy
```

SQL SELECT TOP PERCENT উদাহরণ

নিম্নের SQL `SELECT TOP` স্টেটমেন্টটি "Student_details" টেবিল থেকে প্রথম ৫০% রেকর্ডকে সিলেক্ট করবেঃ

উদাহরণ

```
SELECT TOP 50 PERCENT * FROM Student_details;
```

SQL ORDER BY কিওয়ার্ড

ডেটাবেজ থেকে প্রাপ্ত ফলাফল-সেট কে ঊর্ধ্বক্রম(`ASC`) অথবা অধঃক্রম(`DESC`) অনুসারে সাজাতে SQL `ORDER BY` কিওয়ার্ডটি ব্যবহার করা হয়।

SQL ORDER BY কিওয়ার্ড

`ORDER BY` কিওয়ার্ডটি ব্যবহার করে প্রাপ্ত ফলাফল-সেট কে এক বা একাধিক কলামের উপর ভিত্তিকরে সাজানো যায়। `ORDER BY` কিওয়ার্ডটি তথ্য-সমূহকে ডিফল্টরূপে ঊর্ধ্বক্রমে(`ASC`) সাজায়। কিন্তু আপনি যদি অধঃক্রমে সাজাতে চান তাহলে `DESC` কিওয়ার্ডটি ব্যবহার করতে হবে।

ORDER BY সিনট্যাক্স

```
SELECT name_of_column, name_of_column
FROM name_of_table
ORDER BY name_of_column ASC;
Copy
```

অথবা

```
SELECT name_of_column, name_of_column
FROM name_of_table
ORDER BY name_of_column DESC;
Copy
```

নমুনা ডেটাবেজ

`ORDER BY` কীওয়ার্ডের ব্যবহার দেখার জন্য আমরা আমাদের নমুনা ডেটাবেজ Student ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

ORDER BY উদাহরণ

নিচের SQL `ORDER BY` স্টেটমেন্টি ডেটাবেজের "Student_details" টেবিল থেকে সকল তথ্য সিলেক্ট করে "ঠিকানা(Address)" কলাম অনুসারে ঊর্ধ্বক্রমে অর্থাৎ **Ascending** অর্ডারে সাজাবেঃ

উদাহরণ

```
SELECT * FROM Student_details
ORDER BY Address;
Copy
```

ORDER BY DESC উদাহরণ

নিচের SQL `ORDER BY` স্টেটমেন্টি ডেটাবেজের "Student_details" টেবিল থেকে সকল তথ্য সিলেক্ট করে "ঠিকানা(Address)" কলাম অনুসারে অধঃক্রমে অর্থাৎ **Descending** অর্ডারে সাজাবেঃ

উদাহরণ

```
SELECT * FROM Student_details
ORDER BY Address DESC;
Copy
```

ORDER BY একাধিক কলামের উদাহরণ

নিচের SQL ORDER BY স্টেটমেন্টি ডেটাবেজের "Student_details" টেবিল থেকে সকল তথ্য সিলেক্ট করে "ঠিকানা(Address)" এবং "শিক্ষার্থীর নাম(Student_name)" কলাম অনুসারে ঊর্ধ্বক্রমে অর্থাৎ **Ascending** অর্ডারে সাজাবেঃ

উদাহরণ

```
SELECT * FROM Student_details
ORDER BY Address, Student_name;
Copy
```

ORDER BY একাধিক কলামের জটিল উদাহরণ

নিচের SQL ORDER BY স্টেটমেন্টি ডেটাবেজের "Student_details" টেবিল থেকে সকল তথ্য সিলেক্ট করে "ঠিকানা(Address)" কলামকে ঊর্ধ্বক্রমে অর্থাৎ **Ascending** অর্ডারে এবং "শিক্ষার্থীর নাম(Student_name)" কলামকে অধঃক্রমে অর্থাৎ **Descending** অর্ডারে সাজাবেঃ

উদাহরণ

```
SELECT * FROM Student_details
ORDER BY Address ASC, Student_name DESC;
Copy
```

SQL GROUP BY স্টেটমেন্ট

এক বা একাধিক কলামের রেজাল্ট-সেট কে গ্রুপ করার জন্য Aggregate ফাংশন যেমন- MIN, MAX, AVG, COUNT, SUM ইত্যাদির সাথে প্রায়ই GROUP BY স্টেটমেন্টটি ব্যবহার করা হয়।

GROUP BY স্টেটমেন্ট

GROUP BY স্টেটমেন্ট এর মাধ্যমে এক বা একাধিক কলামের রেজাল্ট-সেট কে একত্রিত(group) করা যায়।

SQL GROUP BY সিনট্যাক্স

```
SELECT aggregate_function(name_of_column), name_of_column
FROM name_of_table
GROUP BY name_of_column's;
Copy
```

উদাহরণ

```
SELECT COUNT(id), Address
FROM Student_details
GROUP BY Address;
Copy
```

নমুনা ডেটাবেজ

GROUP BY স্টেটমেন্টের ব্যবহার দেখার জন্য আমরা আমাদের নমুনা ডেটাবেজ Student ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

নিচের অংশটি "Student_result" টেবিল থেকে নেওয়াঃ

আইডি নং	রোল নাম্বার	ফলাফল
১	১০১	A+
২	১০২	A+
৩	১০৩	A+
৪	১০৪	A+
৫	১০৫	A+

SQL GROUP BY স্টেটমেন্টের উদাহরণ

এখন আমরা প্রতিটি শিক্ষার্থীর ফলাফল খুঁজে বের করবো।

নিম্নলিখিত SQL স্টেটমেন্ট দ্বারা আমরা GROUP BY এর ব্যবহার দেখবোঃ

উদাহরণ

```
SELECT student_details.student_name,student_result.result AS
Result FROM student_result LEFT JOIN student_details
ON student_result.roll_number = student_details.roll_number
GROUP BY student_name;
Copy
```

কিছু গুরুত্বপূর্ণ Aggregate ফাংশন

- **MIN** - একটি প্রদত্ত কলামের সর্বোনিম্ন মান রিটার্ন করে।
- **MAX** - একটি প্রদত্ত কলামের সর্বোচ্চ মান রিটার্ন করে।
- **SUM** - একটি প্রদত্ত কলামে সংখ্যামান-সমূহের যোগফল রিটার্ন করে।
- **AVG** - একটি প্রদত্ত কলামের গড় মান রিটার্ন করে।
- **COUNT** - প্রদত্ত কলামের মান-সমূহের মোট সংখ্যা রিটার্ন করে।
- **COUNT(*)** - একটি টেবিলে মোট সারির সংখ্যা রিটার্ন করে।

SQL SELECT DISTINCT স্টেটমেন্ট

ডেটাবেজে একই তথ্য একাধিকবার থাকলে `SELECT DISTINCT` স্টেটমেন্টটি ব্যবহার করলে ডুপ্লিকেট তথ্য গুলোর শুধুমাত্র একটি আউটপুট আসে।

SQL SELECT DISTINCT স্টেটমেন্ট

ডেটাবেজের টেবিলে একই তথ্য একাধিকবার থাকতে পারে, এই ডুপ্লিকেট তথ্য গুলোকে একক ভাবে পেতে আপনি `SELECT DISTINCT` স্টেটমেন্টটি ব্যবহার করতে পারেন। `SELECT DISTINCT` স্টেটমেন্টটি দ্বারা আপনার ডেটাবেজের অন্তর্ভুক্ত ডুপ্লিকেট তথ্য বাদ দিয়ে স্বতন্ত্রভাবে তালিকাবদ্ধ করতে পারবেন।

`DISTINCT` কিওয়ার্ডটি ব্যবহার করে আপনি শুধুমাত্র স্বতন্ত্র মানগুলো পেতে পারেন।

SQL SELECT DISTINCT স্টেটমেন্ট গঠন

```
SELECT DISTINCT name_of_column's
FROM name_of_table;
```

Collected By
Md. Jubayir Hossain

```
Copy
```

নমুনা ডেটাবেজ

`SELECT DISTINCT` স্টেটমেন্টের ব্যবহার দেখার জন্য আমরা আমাদের নমুনা ডেটাবেজ Student ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

SELECT DISTINCT উদাহরণ

নিম্নের SQL `DISTINCT` স্টেটমেন্টটি "Student_details" টেবিলের "ঠিকানা(Address)" কলামকে সিলেক্ট করবেঃ

উদাহরণ

```
SELECT DISTINCT Address FROM Student_details;
```

SQL SELECT INTO স্টেটমেন্ট

SQL দ্বারা আমরা এক টেবিল থেকে তথ্য অন্য টেবিলে কপি করতে পারি। আপনি SQL `SELECT INTO` স্টেটমেন্ট দ্বারা একটি টেবিল থেকে তথ্য কপি করে নতুন একটি টেবিলে রাখতে পারেন।

SQL SELECT INTO স্টেটমেন্ট

SQL `SELECT INTO` স্টেটমেন্টটি একটি টেবিলের তথ্য কপি করে নতুন একটি টেবিলে রাখে।

SQL SELECT INTO সিনট্যাক্স

আপনি নিম্নের সিনট্যাক্সের মাধ্যমে একটি টেবিলের সকল কলামকে কপি করে একটি নতুন টেবিলের মধ্যে রাখতে পারবেনঃ

```
SELECT *
INTO new_table IN external_database
FROM name_of_table;
Copy
```

অথবা আপনার ইচ্ছামত কলামকে কপি করে নতুন টেবিলে রাখতে পারেনঃ

```
SELECT name_of_column's
INTO new_table IN external_database
FROM name_of_table;
Copy
```

নতুন টেবিলটির কলামের নাম এবং টাইপ `SELECT` স্টেটমেন্টে ডিফাইন করা থাকে। আপনি ইচ্ছা করলে `AS clause` ব্যবহার করে তা পরিবর্তন করতে পারেন।

SQL SELECT INTO উদাহরণ

চলুন "Student_details" টেবিলের একটি ব্যাকআপ কপি তৈরী করিঃ

```
SELECT name_of_column's
INTO Student_details_Backup
FROM Student_details;
Copy
```

`IN clause` ব্যবহার করে টেবিলের তথ্য গুলো অন্য একটি ডেটাবেজে কপি করিঃ

```
SELECT name_of_column's
INTO Student_details_Backup IN 'Backup_database.mdb'
FROM Student_details;
Copy
```

কিছু সংখ্যক কলাম একটি নতুন টেবিলে কপি করিঃ

```
SELECT Student_name, Address
INTO Student_details_Backup
FROM Student_details;
Copy
```

"Student_details" টেবিলের "**ঢাকা**" ঠিকানার অন্তর্ভুক্ত সকল তথ্য নতুন একটি টেবিলে কপি করিঃ

```
SELECT *
INTO Student_details_Backup
FROM Student_details
WHERE Address="ঢাকা";
```

```
Copy
```

একাধিক টেবিলের তথ্য একটি নতুন টেবিলে কপি করিঃ

```
SELECT Student_details.Student_name, Student_result.result
INTO Student_details_Backup
FROM Student_details
LEFT JOIN Student_result
ON Student_details.Roll_number = Student_result.Roll_number;
Copy
```

অন্য একটি টেবিলের মডেল(schema) ব্যবহার করেও `SELECT INTO` স্টেটমেন্ট একটি নতুন ও খালি টেবিল তৈরী করতে পারে। এক্ষেত্রে কুয়েরি করার সময় শুধুমাত্র একটি `WHERE clause` যোগ করলে কোন ডেটা রিটার্ন করবে নাঃ

```
SELECT *
INTO new_table
FROM name_of_table
WHERE 1=0;
```

SQL INSERT INTO SELECT স্টেটমেন্ট

SQL দ্বারা আমরা এক টেবিলের তথ্য অন্য টেবিলে কপি করতে পারি। আপনি SQL `INSERT INTO ... SELECT` স্টেটমেন্টটি ব্যবহার করে এক টেবিলের তথ্য পূর্ব থেকে বিদ্যমান অন্য একটি টেবিলে কপি করে রাখতে পারেন।

SQL INSERT INTO ... SELECT স্টেটমেন্ট

SQL `INSERT INTO ... SELECT` স্টেটমেন্টটি এক টেবিল থেকে তথ্য কপি করে পূর্ব থেকে বিদ্যমান অন্য একটি টেবিলে জমা রাখে। এক্ষেত্রে টার্গেট টেবিলে বিদ্যমান কোনো সারি প্রভাবিত হবে না।

SQL INSERT INTO ... SELECT সিনট্যাক্স

আপনি একটি টেবিলের সকল কলামকে কপি করে পূর্ব থেকে বিদ্যমান একটি টেবিলের মধ্যে রাখতে পারেনঃ

```
INSERT INTO data_table
SELECT * FROM target_table;
Copy
```

অথবা আপনি আপনার ইচ্ছামত কলাম কপি করে পূর্ব থেকে বিদ্যমান টেবিলে রাখতে পারেনঃ

```
INSERT INTO data_table
(name_of_column)
SELECT name_of_column
FROM target_table;
Copy
```

নমুনা ডেটাবেজ

`SELECT DISTINCT` স্টেটমেন্টের ব্যবহার দেখার জন্য আমরা আমাদের নমুনা ডেটাবেজ Student ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

নিচের অংশটি "Teacher_details" টেবিল থেকে নেওয়াঃ

আইডি নং	শিক্ষকের নাম	ঠিকানা
১	মোঃ সাইফুল ইসলাম	ঢাকা
২	ডঃ নিতাই কুমার শাহা	রাজশাহী
৩	মোঃ আবুল কালাম আজাদ	চাঁদপুর
৪	রূপক রায়	চাঁদপুর
৫	নাদিমা আক্তার	ঢাকা

SQL INSERT INTO ... SELECT উদাহরণ

"Teacher_details" টেবিল থেকে কিছু কলামের তথ্য কপি করে "Student_details" টেবিলে রাখিঃ

উদাহরণ

```
INSERT INTO Student_details (Student_name, Address)
SELECT Teacher_name, Address FROM Teacher_details;
Copy
```

শুধুমাত্র "ঢাকা" ঠিকানার(Address) অন্তর্ভুক্ত শিক্ষকের তথ্য কপি করে "Student_details" টেবিলে রাখিঃ

উদাহরণ

```
INSERT INTO Student_details (Student_name, Address)
SELECT Teacher_name, Address FROM Teacher_details
WHERE Address="ঢাকা";
```

SQL NOT NULL কনস্ট্রেইন্ট

একটি টেবিলের ডিফল্ট ভ্যালু `NULL` থাকে।

SQL NOT NULL কনস্ট্রেইন্ট

কোনো ফিল্ডে(কলামে) `NOT NULL` কনস্ট্রেইন্ট(Constraint) সেট করলে সেই ফিল্ড(কলাম) ফাঁকা(`NULL`) ভ্যালু গ্রহণ করে না। সুতরাং `NOT NULL` কনস্ট্রেইন্টটি সংশ্লিষ্ট ফিল্ডে ভ্যালু রাখতে বাধ্য করে। অর্থাৎ আপনি টেবিলে কোন তথ্য ইনসার্ট অথবা আপডেট করতে চাইলে ঐ ফিল্ডে অবশ্যই ভ্যালু ইনসার্ট করতে হবে, অন্যথায় টেবিলে কোন তথ্য ইনসার্ট অথবা আপডেট করতে পারবেন না।

নিম্নের SQL কনস্ট্রেইন্টটি "Roll_number" এবং "Student_name" কলামে `NULL` ভ্যালু গ্রহণ করবে নাঃ

উদাহরণ

```
CREATE TABLE Student_NotNull(
Id int auto_increment,
Roll_number varchar(255) NOT NULL,
Student_name varchar(255) NOT NULL,
Address varchar(255)
);
Copy
```

উপরের তৈরিকৃত টেবিলের "Roll_number" এবং "Student_name" কলামে তথ্য ইনপুট করা ব্যাতিত সম্পূর্ণ টেবিলে কোন তথ্য ইনপুট বা আপডেট করা যাবে না।

SQL UNIQUE কনস্ট্রেইন্ট

`UNIQUE` কনস্ট্রেইন্ট(Constraint) দ্বারা ডেটাবেজে ইউনিক(Unique) রেকর্ড ইনসার্ট করা হয়। এক বা একাধিক কলামের রেকর্ডকে ইউনিক করতে `UNIQUE` এবং `PRIMARY KEY` কনস্ট্রেইন্ট উভই ব্যবহার করা হয়। একটি `PRIMARY KEY` কনস্ট্রেইন্ট এর মধ্যে স্বয়ংক্রিয়ভাবে `UNIQUE` কনস্ট্রেইন্ট ডিফাইন করা থাকে।

মনে রাখবেন, একটি টেবিলের একাধিক কলামে UNIQUE কনস্ট্রেইন্ট থাকতে পারে কিন্তু একটি টেবিলের শুধুমাত্র একটি কলামেই PRIMARY KEY কনস্ট্রেইন্ট ব্যবহার করা যায়।

টেবিল তৈরির সময় SQL UNIQUE কনস্ট্রেইন্ট এর ব্যবহার

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিল তৈরির সময় "Roll_number" কলামটি UNIQUE কনস্ট্রেইন্ট এর মাধ্যমে ইউনিক হবে।

MySQL এর জন্যঃ

```
CREATE TABLE Student_details(
        Id int NOT NUll,
    Roll_number varchar(255),
    Student_name varchar(255),
    Institute varchar(255),
    Address varchar(255),
    UNIQUE(roll_number)
);
Copy
```

Oracle/SQL Server/MS Access এর জন্যঃ

```
CREATE TABLE Student_details(
        Id int NOT NUll,
    Roll_number varchar(255) UNIQUE,
    Student_name varchar(255),
    Institute varchar(255),
    Address varchar(255)
);
Copy
```

একটি UNIQUE কনস্ট্রেইন্ট এর নাম দেওয়ার জন্য এবং একাধিক কলামে UNIQUE কনস্ট্রেইন্ট ডিফাইন করার জন্য নিম্নের SQL সিনট্যাক্সটি ব্যবহার করা হয়।

MySQL/Oracle/SQL Server/MS Access এর জন্যঃ

```
CREATE TABLE Student_details(
        Id int NOT NUll,
  Roll_number varchar(255),
  Student_name varchar(255),
  Institute varchar(255),
  Address varchar(255),
CONSTRAINT Student_id UNIQUE (Id, Roll_number)
);
Copy
```

পূর্বের তৈরি টেবিলে SQL UNIQUE কনস্ট্রেইন্টের ব্যবহার

ডেটাবেজে পূর্ব থেকে বিদ্যমান একটি টেবিলের "Id" কলামকে `UNIQUE` করতে `ALTER TABLE` স্টেটমেন্টটের সাথে নিম্নের ন্যায় `ADD UNIQUE` কনস্ট্রেইন্ট ব্যবহার করা হয়ঃ

MySQL/Oracle/SQL Server/MS Access এর জন্যঃ

```
ALTER TABLE Student_details
ADD UNIQUE (Id);
Copy
```

একটি `UNIQUE` কনস্ট্রেইন্ট এর নাম দেওয়ার জন্য এবং একাধিক কলামে `UNIQUE` কনস্ট্রেইন্ট ডিফাইন করার জন্য নিম্নের SQL সিনট্যাক্সটি ব্যবহার করা হয়।

MySQL/Oracle/SQL Server/MS Access এর জন্যঃ

```
ALTER TABLE Student_details
ADD CONSTRAINT Student_id UNIQUE (Id, Roll_number);
Copy
```

UNIQUE কনস্ট্রেইন্ট ডিলেট করা

`UNIQUE` কনস্ট্রেইন্ট ডিলেট করতে নিম্নের SQL স্টেটমেন্টটি ব্যবহার করা হয়ঃ

MySQL এর জন্যঃ

```
ALTER TABLE Student_details
DROP INDEX Student_id;
Copy
```

Oracle/SQL Server/MS Access এর জন্যঃ

```
ALTER TABLE Student_details
DROP CONSTRAINT Student_id;
Copy
```

SQL Default কনস্ট্রেইন্ট

Default কনস্ট্রেইন্ট(Constraint) ব্যবহার করে কলামের মধ্যে একটি ডিফল্ট(Default) ভ্যালু ইনপুট করা যায়। কলামে কোন নির্দিষ্ট মান ইনপুট না করা হলে সকল নতুন রেকর্ডের মধ্যে স্বয়ংক্রিয়ভাবে Default ভ্যালুটি যোগ হয়ে যায়।

টেবিল তৈরির ক্ষেত্রে SQL Default কনস্ট্রেইন্টের ব্যবহার

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিল তৈরী করার সময় "Institute" কলামে Default কনস্ট্রেইন্ট সেট করবেঃ

MySQL /Oracle/ SQL Server/MS Access এর জন্যঃ

```
CREATE TABLE Student_details(
        Id int NOT NUll,
    Roll_number varchar(255),
    Student_name varchar(255),
    Institute varchar(255) DEFAULT 'জাতীয় বিশ্ববিদ্যালয়',
    Address varchar(255)
);
Copy
```

সিস্টেম ভ্যালু ইনর্সাট করার জন্য ফাংসশন যেমন-GETDATE() এর সাথেও **Default** কনস্ট্রেইন্ট ব্যবহার করা হয়ঃ

```
CREATE TABLE Student_attendance(
        Id int NOT NUll,
    Roll_number varchar(255),
    Attendance varchar(255),
    Admission_date date DEFAULT GETDATE()
);
Copy
```

পূর্বের তৈরি টেবিলে SQL Default কনস্ট্রেইন্টের ব্যবহার

নিম্নের SQL স্টেটমেন্টটি পূর্বে তৈরিকৃত "Student_details" টেবিলের "Institute" কলামে Default কনস্ট্রেইন্ট সেট করবেঃ

MySQL এর জন্যঃ

```
ALTER TABLE Student_details
ALTER Institute SET DEFAULT 'জাতীয় বিশ্ববিদ্যালয়';
Copy
```

Oracle এর জন্যঃ

```
ALTER TABLE Student_details
MODIFY Institute DEFAULT 'জাতীয় বিশ্ববিদ্যালয়';
Copy
```

SQL Server/MS Access এর জন্যঃ

```
ALTER TABLE Student_details
ALTER COLUMN Institute SET DEFAULT 'জাতীয় বিশ্ববিদ্যালয়';
Copy
```

Default কনস্ট্রেইন্ট ডিলেট করা

একটি `Default` কনস্ট্রেইন্ট ডিলিট করতে নিম্নের SQL স্টেটমেন্টটি ব্যবহার করুনঃ

MySQL এর জন্যঃ

```
ALTER TABLE Student_details
ALTER Institute DROP DEFAULT;
Copy
```

Oracle/SQL Server/MS Access এর জন্যঃ

```
ALTER TABLE Student_details
ALTER COLUMN Institute DROP DEFAULT;
Copy
```

SQL CHECK কনস্ট্রেইন্ট

একটি কলাম কতটি ভ্যালু গ্রহণ করবে তার রেঞ্জ নির্ধারন করতে `CHECK` কনস্ট্রেইন্ট(Constraint) ব্যবহার করা হয়। আপনি যদি একটি নির্দিষ্ট কলামের জন্য `CHECK` কনস্ট্রেইন্ট নির্ধারন করে দেন তাহলে এটি শুধুমাত্র ঐ কলামের ভ্যালুর জন্যই প্রযোজ্য হবে। আপনি যদি একটি টেবিলের জন্য কনস্ট্রেইন্ট ডিফাইন করে থাকেন তাহলে তা টেবিলে অন্তর্ভূক্ত সকল কলামের জন্যই প্রযোজ্য হবে।

SQL টেবিল তৈরীর সময় CHECK কনস্ট্রেইন্টের ব্যবহার

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিল তৈরী করার সময় "Id" কলামে CHECK কনস্ট্রেইন্ট সেট করবে। এক্ষেত্রে CHECK কনস্ট্রেইন্ট নির্ধারন করে দিবে যেন "Id" কলামের ভ্যালু-সমূহ শূণ্য(0) থেকে বড় হয়।

MySQL এর জন্যঃ

```
CREATE TABLE Student_details(
  Id int NOT NUll,
  Roll_number varchar(255),
  Student_name varchar(255),
  Institute varchar(255),
  Address varchar(255),
CHECK (Id>0)
);
Copy
```

Oracle/SQL Server/MS Acces এর জন্যঃ

```
CREATE TABLE Student_details(
  Id int NOT NUll CHECK (Id>0),
  Roll_number varchar(255),
  Student_name varchar(255),
  Institute varchar(255),
  Address varchar(255)
);
Copy
```

একটি CHECK কনস্ট্রেইন্ট এর নাম দেওয়ার জন্য এবং একাধিক কলামে CHECK কনস্ট্রেইন্ট ডিফাইন করার জন্য নিম্নের SQL সিনট্যাক্সটি ব্যবহার করা হয়।

MySQL / SQL Server / Oracle / MS Access এর জন্যঃ

```
CREATE TABLE Student_details(
  Id int NOT NUll,
  Roll_number varchar(255),
  Student_name varchar(255),
  Institute varchar(255),
  Address varchar(255),
  CONSTRAINT check_student CHECK (Id>0 AND Address='জাতীয় বিশ্ববিদ্যালয়')
);
Copy
```

পূর্বের তৈরি টেবিলে SQL UNIQUE কনস্ট্রেইন্ট এর ব্যবহার

ডেটাবেজে পূর্ব থেকে বিদ্যমান একটি টেবিলের "Id" কলামে CHECK সেট করতে ALTER TABLE স্টেটমেন্টটের সাথে নিম্নের ন্যায় ADD CHECK কনস্ট্রেইন্ট ব্যবহার করা হয়ঃ

MySQL/Oracle/SQL Server/MS Access এর জন্যঃ

```
ALTER TABLE Student_details
ADD CHECK (Id>0);
Copy
```

একটি CHECK কনস্ট্রেইন্ট এর নাম দেওয়ার জন্য এবং একাধিক কলামে CHECK কনস্ট্রেইন্ট ডিফাইন করার জন্য নিম্নের SQL সিনট্যাক্সটি ব্যবহার করা হয়।

MySQL/Oracle/SQL Server/MS Access এর জন্যঃ

```
ALTER TABLE Student_details
ADD CONSTRAINT check_student CHECK (Id>0 AND Address='জাতীয় বিশ্ববিদ্যালয়');
Copy
```

একটি CHECK কনস্ট্রেইন্ট ডিলেট করা

একটি CHECK কনস্ট্রেইন্ট ডিলেট করতে, নিম্নের SQL সিনট্যাক্সটি ব্যবহার করা হয়ঃ

MySQL এর জন্যঃ

```
ALTER TABLE Student_details
DROP CHECK check_student;
Copy
```

MySQL/Oracle/SQL Server/MS Access এর জন্যঃ

```
ALTER TABLE Student_details
DROP CONSTRAINT check_student;
```

SQL PRIMARY KEY কনস্ট্রেইন্ট

PRIMARY KEY কনস্ট্রেইন্ট(Constraint) ডেটাবেজ টেবিলের প্রতিটি রেকর্ডকে ইউনিকভাবে শনাক্ত করে। Primary key এর ভ্যালু-সমূহ অবশ্যই ইউনিক হতে হবে।

primary key বিশিষ্ট কলামে NULL ভ্যালু থাকতে পারবে না। অশিকাংশ টেবিলেই primary key থাকা উচিৎ এবং প্রত্যেক টেবিলে শুধুমাত্র একটি primary key থাকতে পারবে।

টেবিল তৈরির সময় SQL PRIMARY KEY কনস্ট্রেইন্ট এর ব্যবহার।

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিল তৈরির সময় "Id" কলামে `PRIMARY KEY` কনস্ট্রেইন্ট সেট করবেঃ

MySQL এর জন্যঃ

```
CREATE TABLE Student_details(
        Id int NOT NUll,
    Roll_number varchar(255),
    Student_name varchar(255),
    Institute varchar(255),
    Address varchar(255),
    PRIMARY KEY (Id)
);
Copy
```

Oracle/SQL Server/MS Access এর জন্যঃ

```
CREATE TABLE Student_details(
        Id int NOT NUll PRIMARY KEY,
    Roll_number varchar(255),
    Student_name varchar(255),
    Institute varchar(255),
    Address varchar(255)
);
Copy
```

একটি `PRIMARY KEY` কনস্ট্রেইন্ট এর নাম দেওয়ার জন্য এবং একাধিক কলামে `PRIMARY KEY` কনস্ট্রেইন্ট ডিফাইন করার জন্য নিম্নের SQL সিনট্যাক্সটি ব্যবহার করা হয়ঃ

MySQL/Oracle/SQL Server/MS Access এর জন্যঃ

```
CREATE TABLE Student_details(
        Id int NOT NUll PRIMARY KEY,
    Roll_number varchar(255),
    Student_name varchar(255),
    Institute varchar(255),
    Address varchar(255),
    CONSTRAINT Student_Id PRIMARY KEY (Id,Roll_number)
);
Copy
```

বিঃদ্রঃ উপরের উদাহরণে শুধুমাত্র একটি `PRIMARY KEY`(Student_Id) রয়েছে। যদিও `primary key` ভ্যালুটি দুইটি কলামের(Id + Roll_number) উপর ভিত্তি করে গঠিত।

পূর্বের তৈরি টেবিলে SQL PRIMARY KEY কনস্ট্রেইন্ট এর ব্যবহার।

ডেটাবেজে পূর্ব থেকে বিদ্যমান একটি টেবিলের "Id" কলামে `PRIMARY KEY` সেট করতে `ALTER TABLE` স্টেটমেন্টটের সাথে নিম্নের ন্যায় `add PRIMARY KEY` কনস্ট্রেইন্ট ব্যবহার করা হয়ঃ

MySQL/Oracle/SQL Server/MS Access এর জন্যঃ

```
ALTER TABLE Student_details(
ADD PRIMARY KEY (Id);
Copy
```

একটি `PRIMARY KEY` কনস্ট্রেইন্ট এর নাম দেওয়ার জন্য এবং একাধিক কলামে `PRIMARY KEY` কনস্ট্রেইন্ট ডিফাইন করার জন্য নিম্নের SQL সিনট্যাক্সটি ব্যবহার করা হয়।

মাMySQL/Oracle/SQL Server/MS Access এর জন্যঃ

```
ALTER TABLE Student_details(
ADD CONSTRAINT Student_Id PRIMARY KEY (Id,Roll_number);
Copy
```

বিঃদ্রঃ `ALTER TABLE` স্টেটমেন্ট ব্যবহার করে কোনো কলামে `primary key` যোগ করতে হলে টেবিল তৈরির সময়ে ঐ কলামকে অবশ্যই **প্রাইমারি কী কলাম** হিসাবে ডিক্লেয়ার করতে হবে যেন সে `NULL` ভ্যালু গ্রহণ করতে না পারে।

PRIMARY KEY কনস্ট্রেইন্ট ডিলেট করা

`PRIMARY KEY` কনস্ট্রেইন্ট ডিলেট করতে নিম্নের SQL স্টেটমেন্টটি ব্যবহার করা হয়ঃ

MySQL এর জন্যঃ

```
ALTER TABLE Student_details(
DROP PRIMARY KEY;
Copy
```

Oracle/SQL Server/MS Access এর জন্যঃ

```
ALTER TABLE Student_details(
DROP CONSTRAINT Student_Id;
```

QL FOREIGN KEY কনস্ট্রেইন্ট

এক টেবিলের `FOREIGN KEY` অন্য টেবিলের `PRIMARY KEY` কে নির্দেশ করে। সুতরাং দুটি টেবিলের মধ্যে সংযোগ সৃষ্টি করে।

দুটি টেবিলের সংযোগ বিচ্ছিন্ন করে এমন ক্রিয়া কলাপে `FOREIGN KEY` কনস্ট্রেইন্ট বাধা দেয়।

`FOREIGN KEY` কনস্ট্রেইন্ট **ফরেন কী কলামে** অবৈধ ডেটা ইনপুটেও বাধা দেয়, কারন **প্রাইমারি কী কলাম** এর ভ্যালুই হলো **ফরেন কি কলাম** এর ভ্যালু।

চলুন একটি উদাহরণের সাহায্য `FOREIGN KEY` বুঝার চেষ্টা করি। নিম্নের টেবিল দুটিতে ভালভাবে লক্ষ্য করুনঃ

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

নিচের অংশটি "Student_result" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	ফলাফল
১	১০১	A+
২	১০২	A+
৩	১০৩	A+
৪	১০৪	A+
৫	১০৫	A+

উপরের টেবিল দুটিতে লক্ষ্য করলে দেখবেন, "Student_result" টেবিলের "Roll_number" কলামটি "Student_details" টেবিলের "Roll_number" কলামকে নির্দেশ(Point) করছে।

"Student_details" টেবিলের "Roll_number" কলামটি হচ্ছে ঐ টেবিলের `PRIMARY KEY` কনস্ট্রেইন্ট এবং "Student_result" টেবিলের "Roll_number" কলামটি হচ্ছে "Student_result" টেবিলের `FOREIGN KEY` কনস্ট্রেইন্ট।

বিঃদ্রঃ `PRIMARY KEY` এবং `FOREIGN KEY` কে যথাক্রমে **পিতা** এবং **পুত্র** বলা হয়। পুত্রের কোনো কিছু প্রয়োজন হলে সে পিতার কাছেই চাই। একইভাবে `FOREIGN KEY` তার প্রয়োজনে `PRIMARY KEY` এর কাছে রেফার করে।

টেবিল তৈরির সময় FOREIGN KEY কনস্ট্রেইন্ট সেট করা

নিম্নের SQL স্টেটমেন্টটি "Student_result" টেবিল তৈরি করার সময় "Roll_number" কলামে একটি `FOREIGN KEY` কনস্ট্রেইন্ট সেট করবেঃ

MySQL এর জন্যঃ

```
CREATE TABLE Student_result(
        Id int NOT NUll,
    Roll_number varchar(255) NOT NULL,
    Result varchar(255) NOT NULL,
    PRIMARY KEY (Id),
    FOREIGN KEY (Roll_number) REFERENCES Student_details(Roll_number)
);
Copy
```

Oracle/SQL Server/MS Access এর জন্যঃ

```
CREATE TABLE Student_result(
        Id int NOT NUll,
    Roll_number varchar(255) FOREIGN KEY REFERENCES Student_details(Roll_number),
    Result varchar(255) NOT NULL
);
Copy
```

`FOREIGN KEY` কনস্ট্রেইন্ট এর নামকরন এবং একাধিক কলামে `FOREIGN KEY` কনস্ট্রেইন্ট ডিফাইন করতে নিম্নের SQL স্টেটমেন্টটি ব্যবহার করা হয়ঃ

MySQL/Oracle/SQL Server/MS Access এর জন্যঃ

```
CREATE TABLE Student_result(
        Id int NOT NUll,
    Roll_number varchar(255) NOT NULL,
    Result varchar(255) NOT NULL,
    PRIMARY KEY (Id),
    CONSTRAINT link_Student_result FOREIGN KEY(Roll_number) REFERENCES
Student_details(Roll_number)
);
Copy
```

টেবিল পরিবর্তন করার সময় FOREIGN KEY সেট করা

নিম্নের SQL স্টেটমেন্টটি "Student_result" টেবিল পরিবর্তন(modify) করার সময় "Roll_number" কলামে `FOREIGN KEY` কনস্ট্রেইন্ট সেট করবেঃ

MySQL/Oracle/SQL Server/MS Access এর জন্যঃ

```
ALTER TABLE Student_result
ADD FOREIGN KEY (Roll_number)
REFERENCES Student_details(Roll_number);
Copy
```

`FOREIGN KEY` কনস্ট্রেইন্ট এর নামকরন এবং একাধিক কলামে `FOREIGN KEY` কনস্ট্রেইন্ট ডিফাইন করতে নিম্নের SQL স্টেটমেন্টটি ব্যবহার করা হয়ঃ

MySQL/Oracle/SQL Server/MS Access এর জন্যঃ

```
ALTER TABLE Student_result
ADD CONSTRAINT link_Student_result
ADD FOREIGN KEY (Roll_number)
REFERENCES Student_details(Roll_number);
Copy
```

FOREIGN KEY কনস্ট্রেইন্ট ডিলেট

`FOREIGN KEY` কনস্ট্রেইন্টকে ডিলেট করতে নিম্নের SQL স্টেটমেন্টটি ব্যবহার করুনঃ

MySQL এর জন্যঃ

```
ALTER TABLE Student_result
DROP FOREIGN KEY link_Student_result;
Copy
```

Oracle/SQL Server/MS Access এর জন্যঃ

```
ALTER TABLE Student_result
DROP CONSTRAINT link_Student_result;
```

SQL IN অপারেটর

`IN` অপারেটর দ্বারা `WHERE clause` এর মধ্যে একত্রে একাধিক ভ্যালু উল্লেখ করা যায়।

SQL IN সিনটেক্স

```
SELECT name_of_column's
FROM name_of_table
WHERE name_of_column IN (value_1, value_2,...);
Copy
```

নমুনা ডেটাবেজ

IN অপারেটর এর ব্যবহার দেখার জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

IN অপারেটরের উদাহরণ

নিম্নের SQL IN স্টেটমেন্টটি "Student_details" টেবিল থেকে "ঢাকা" অথবা "রাজশাহী" শহরের রেকর্ডকে সিলেক্ট করবেঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE Address IN ('ঢাকা','রাজশাহী');
```

SQL BETWEEN অপারেটর

একটি নির্দিষ্ট রেঞ্জের মধ্যে ভ্যালু সিলেক্ট করার জন্য SQL **BETWEEN** অপারেটর ব্যবহার করা হয়।

SQL BETWEEN অপারেটর

BETWEEN অপারেটর একটি নির্দিষ্ট রেঞ্জের ভ্যালু সিলেক্ট করে। এই ভ্যালু-সমূহ টেক্সট, সংখ্যা অথবা তারিখ হতে পারে।

SQL BETWEEN সিনট্যাক্স

```
SELECT name_of_column's
FROM name_of_table
```

```
WHERE name_of_column BETWEEN value_1 AND value_2;
Copy
```

নমুনা ডেটাবেজ

IN অপারেটর এর ব্যবহার দেখার জন্য আমরা আমাদের **নমুনা** ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

BETWEEN অপারেটরের উদাহরণ

নিম্নের SQL **BETWEEN** স্টেটমেন্টটি "Student_details" টেবিলের "Roll_number" কলামের ১০১ থেকে ১০৫ রেঞ্জ পর্যন্ত সকল রেকর্ড গুলো নিয়ে আসবেঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE Roll_number BETWEEN '১০১' AND '১০৫';
Copy
```

সতর্কতাঃ BETWEEN অপারেটরটি বিভিন্ন ডেটাবেজে বিভিন্ন ফলাফল দেখাতে পারে!

- কিছু ডেটাবেজে BETWEEN অপারেটরটি টেষ্ট ভ্যালুসহ এর মাঝখানের ভ্যালুগুলোর জন্য ফলাফল দেয়।
- আবার কিছু ডেটাবেজে টেষ্ট ভ্যালুছাড়া এর মাঝখানের ভ্যালুগুলোর জন্য ফলাফল দেয়।
- আবার কিছু ডেটাবেজে প্রথম টেষ্ট ভ্যালুসহ এবং শেষ টেষ্ট ভ্যালু ছাড়া এর মাঝখানের ভ্যালুগুলোর জন্য ফলাফল দেয়।

সুতরাং BETWEEN নিয়ে কাজ করার পূর্বে BETWEEN অপারেটরটি আপনার ডেটাবেজে কিভাবে কাজ করে তা চেক করে দেখুন!

NOT BETWEEN অপারেটরের উদাহরণ

পূর্ববর্তী উদাহরণে আমরা রেঞ্জের ভিতরের ভ্যালু-সমূহ সিলেক্ট করেছিলাম। আর এই উদাহরণে আমরা ঐ নির্দিষ্ট রেঞ্জের বাইরের ভ্যালু-সমূহ সিলেক্ট করবো। এজন্য আমরা **BETWEEN** এর পরিবর্তে **NOT BETWEEN** ব্যবহার করবো।

উদাহরণ

```
SELECT * FROM Student_details
WHERE Roll_number NOT BETWEEN '১০১' AND '১০৫';
Copy
```

BETWEEN এবং IN অপারেটর একত্রে ব্যবহার করা

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিলের "Roll_number" কলামের ১০১ থেকে ১১০ রেঞ্জ পর্যন্ত রেকর্ড গুলো নিয়ে আসবে, কিন্তু ৪, ৭ এবং ৯ "Id" বিশিষ্ট রেকর্ড গুলো দেখাবে নাঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE (Roll_number BETWEEN '১০১' AND '১১০')
AND NOT Id IN(৪, ৭, ৯);
Copy
```

টেক্সট ভ্যালুসহ BETWEEN অপারেটরের ব্যবহার

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিল এর রেকর্ড-সমূহের মধ্যে যে সকল "শিক্ষার্থীর নাম(Student_name)" 'ক' থেকে 'ম' পর্যন্ত অক্ষর দ্বারা শুরু হয়েছে শুধুমাত্র তাদেরকে নিয়ে আসবেঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE Student_name BETWEEN 'ক' AND 'ম';
Copy
```

টেক্সট ভ্যালুসহ NOT BETWEEN অপারেটরের ব্যবহার

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিল এর রেকর্ড-সমূহের মধ্যে যে সকল "শিক্ষার্থীর নাম(Student_name)" 'ক' থেকে 'ম' পর্যন্ত অক্ষর দ্বারা শুরু হয়েছে শুধুমাত্র তাদেরকে ছাড়া বাকীদেরকে নিয়ে আসবেঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE Student_name NOT BETWEEN 'ক' AND 'ম';
Copy
```

নমুনা ডেটাবেজ

নিচের অংশটি "Student_attendance" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শতকরা উপস্থিতি	ভর্তির তারিখ
১	১০১	৮৯%	০১-১১-২০১৫
২	১০২	৯১%	০১-১১-২০১৫
৩	১০৩	৮০%	০১-১১-২০১৫
৪	১০৪	৭৫%	০২-১১-২০১৫
৫	১০৫	৭৭%	০২-১১-২০১৫

ডেট ভ্যালুসহ BETWEEN অপারেটরের ব্যবহার

নিম্নের SQL স্টেটমেন্টটি "Student_attendance" টেবিল এর রেকর্ড-সমূহের মধ্যে যে সকল শিক্ষার্থীর "ভর্তির তারিখ(Admission_date)" '০১-১১-২০১৫' থেকে '১০-১১-২০১৫' এর মধ্যে শুধুমাত্র তাদেরকে নিয়ে আসবেঃ

উদাহরণ

```
SELECT * FROM Student_attendance
WHERE Admission_date BETWEEN '০১-১১-২০১৫' AND '১০-১১-২০১৫';
```

SQL AUTO INCREMENT এট্রিবিউট

যখন কোনো টেবিলের মধ্যে নতুন তথ্য যোগ করা হবে তখন `Auto-increment` এট্রিবিউটটি একটি নির্দিষ্ট কলামের জন্য একটি ইউনিক নম্বর তৈরি করবে।

SQL ফিল্ড AUTO INCREMENT

Auto-increment এট্রিবিউটটি ডেটাবেজ টেবিলের প্রত্যেক নতুন সারির জন্য একটি ইউনিক আইডেন্টিটি জেনারেট করে। ডেটাবেজ টেবিলে তথ্য ইনপুট করার সময় আপনি এমনটা চাইতেই পারেন যে, প্রত্যেক সারি ইনপুটের জন্য একটি ইউনিক নম্বর তৈরি হোক।

এক্ষেত্রে আপনি একটি auto-increment ফিল্ড(column) তৈরি করতে পারেন। আপনার নতুন তথ্য ইনপুট করার সাথে সাথে প্রত্যেক সারি ইনপুটের জন্য এটি স্বয়ংক্রিয়ভাবে একটি ইউনিক নম্বর জেনারেট করবে।

প্রতি ফিল্ডের ক্ষেত্রে এটি স্বয়ংক্রিয়ভাবে বৃদ্ধি পাবে। আর আমরা বেশীরভাগ সময়েই চাই টেবিলে নতুন তথ্য যোগ হওয়ার সাথে সাথে primary key এর মান স্বয়ংক্রিয়ভাবে তৈরি হয়ে যাক।। টেবিলের মধ্যে auto-increment ফিল্ড তৈরি করার মাধ্যমে আমরা এটি করতে পারি।

MySQL সিনট্যাক্স

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিলের "Id" কলামকে primary key ফিল্ডে রূপান্তর করবে এবং এটি auto-increment হবেঃ

```
CREATE TABLE Student_details(
  Id int NOT NUll AUTO_INCREMENT,
  Roll_number varchar(255),
  Student_name varchar(255),
  Institute varchar(255),
  Address varchar(255),
  PRIMARY KEY (Id)
);
Copy
```

MySQL এ কোনো ফিল্ডের মান স্বয়ংক্রিয়ভাবে বৃদ্ধি করার জন্য AUTO_INCREMENT কিওয়ার্ডটি ব্যবহার করা হয়। ডিফল্টভাবে AUTO_INCREMENT এর মান ১ থেকে শুরু হয় এবং প্রতিটি নতুন রেকর্ডের জন্য ১ করে বৃদ্ধি পায়।

যদি AUTO_INCREMENT এর মান অন্য কোনো মান দ্বারা শুরু করতে চান তাহলে নিম্নের SQL স্টেটমেন্টটি ব্যবহার করতে হবেঃ

```
ALTER TABLE Student_details AUTO_INCREMENT=100;
Copy
```

যখন আমরা "Student_details" টেবিলে তথ্য ইনপুট করবো তখন "Id" কলামের জন্য কোনো তথ্য ইনপুট করবো না, কারন এর জন্য একটি ইউনিক মান স্বয়ংক্রিয়ভাবে যুক্ত হয়ে যাবেঃ

```
INSERT INTO Student_details (Roll_number,Student_name)
VALUES ('১৩১','শাহরিয়ার হাসান');
Copy
```

উপরের SQL স্টেটমেন্টটি "Student_details" টেবিলে একটি নতুন তথ্য ইনপুট করবে।
"Id" কলামে স্বয়ংক্রিয়ভাবেই একটি ইউনিক মান যুক্ত হয়ে যাবে। "রোল নাম্বার(Roll_number)" কলামে "১৩১" এবং "শিক্ষার্থীর নাম(Student_name)" কলামে "শাহরিয়ার হাসান" মান যুক্ত করবে।

SQL Server এর সিনট্যাক্স

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিলের "Id" কলামকে `primary key` ফিল্ডে রূপান্তর করবে এবং এটি `auto-increment` হবেঃ

```
CREATE TABLE Student_details(
  Id int IDENTITY(1,1) PRIMARY KEY,
  Roll_number varchar(255),
  Student_name varchar(255),
  Institute varchar(255),
  Address varchar(255)
);
Copy
```

MS SQL Server-এ কোনো ফিল্ডের মান স্বয়ংক্রিয়ভাবে বৃদ্ধি করার জন্য `IDENTITY` কিওয়ার্ডটি ব্যবহার করা হয়। ডিফল্টভাবে `IDENTITY` এর মান ১ থেকে শুরু হয় এবং প্রতিটি নতুন রেকর্ডের জন্য ১ করে বৃদ্ধি পায়।

বিঃদ্রঃ আপনি "Id" কলামের শুরুর মান এবং বৃদ্ধি হওয়ার মান নির্দিষ্ট করে দেওয়ার জন্য নিম্নের সিনট্যাক্সটি ব্যবহার করতে পারেনঃ

```
IDENTITY(শুরুর মান, বৃদ্ধির মান)
Copy
```

আমরা যখন "Student_details" টেবিলে তথ্য ইনপুট করবো তখন "Id" কলামের জন্য কোনো তথ্য ইনপুট করবো না, কারন এর জন্য টেবিলে একটি ইউনিক মান স্বয়ংক্রিয়ভাবে যুক্ত হয়ে যাবেঃ

```
INSERT INTO Student_details (Roll_number,Student_name)
VALUES ('১৩১','শাহরিয়ার হাসান');
Copy
```

উপরের SQL স্টেটমেন্টটি "Student_details" টেবিলে একটি নতুন তথ্য ইনপুট করবে। "Id" কলামে একটি মান স্বয়ংক্রিয়ভাবে যুক্ত হয়ে যাবে। আর "রোল নাম্বার(Roll_number)" কলামে "১৩১" এবং "শিক্ষার্থীর নাম(Student_name)" কলামে "শাহরিয়ার হাসান" মান যুক্ত করবে।

MS Access এর সিনট্যাক্স

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিলের "Id" কলামকে `primary key` ফিল্ডে রূপান্তর করবে এবং এটি `auto-increment` হবেঃ

```
CREATE TABLE Student_details(
  Id Integer PRIMARY KEY AUTOINCREMENT,
  Roll_number varchar(255),
  Student_name varchar(255),
  Institute varchar(255),
  Address varchar(255)
);
Copy
```

MS Access -এ কোনো ফিল্ডের মান স্বয়ংক্রিয়ভাবে বৃদ্ধি করার জন্য `AUTOINCREMENT` কিওয়ার্ডটি ব্যবহার করা হয়। ডিফল্টভাবে `AUTOINCREMENT` এর মান ১ থেকে শুরু হয় এবং প্রতিটি নতুন রেকর্ডের জন্য ১ করে বৃদ্ধি পায়।

বিঃদ্রঃ আপনি "Id" কলামের শুরুর মান এবং বৃদ্ধি হওয়ার মান নির্দিষ্ট করে দেওয়ার জন্য নিম্নের সিন্ট্যাক্সটি ব্যবহার করতে পারেনঃ

```
IDENTITY(শুরুর মান, বৃদ্ধির মান)
Copy
```

আমরা যখন "Student_details" টেবিলে তথ্য ইনপুট করবো তখন "Id" কলামের জন্য কোনো তথ্য ইনপুট করবো না, কারন এর জন্য টেবিলে একটি ইউনিক মান স্বয়ংক্রিয়ভাবে যুক্ত হয়ে যাবেঃ

```
INSERT INTO Student_details (Roll_number,Student_name)
VALUES ('১৩১','শাহরিয়ার হাসান');
Copy
```

উপরের SQL স্টেটমেন্টটি "Student_details" টেবিলে একটি নতুন তথ্য ইনপুট করবে। "Id" কলামে স্বয়ংক্রিয়ভাবেই একটি ইউনিক মান যুক্ত হয়ে যাবে। "রোল নাম্বার(Roll_number)" কলামে "১৩১" এবং "শিক্ষার্থীর নাম(Student_name)" কলামে "শাহরিয়ার হাসান" মান যুক্ত করবে।

Oracle এর সিনট্যাক্স

উপরের কোড থেকে ওরাকলের কোড একটু ভিন্ন। **সিকুয়েন্স অবজেক্টের** মাধ্যমে আপনি একটি `auto-increment` ফিল্ড তৈরি করতে পারেন। এই অবজেক্টটি নম্বরের একটি **সিকুয়েন্স অবজেক্ট** তৈরি করে।

নিম্নে `CREATE SEQUENCE` এর সিনট্যাক্স দেওয়া হলঃ

```
CREATE SEQUENCE Student_id
MINVALUE 1
START WITH 1
INCREMENT BY 1
CACHE 10
```

```
Copy
```

উপরের কোডটি "Student_id" নামে একটি **সিকুয়েন্স অবজেক্ট** তৈরি করবে। যাহা ১ থেকে শুরু হবে এবং এক এক করে বৃদ্ধি পাবে। এছাড়া এটি পারফরম্যান্সের জন্য ১০ টি পর্যন্ত মান ক্যাশ(cache) করে রাখবে। মেমরি দ্রুত অ্যাক্সেস করার জন্য ক্যাশ অপসনটি মেমোরিতে জমাকৃত মান নির্ধারন করে। ।

এখন আমরা "Student_details" টেবিলে নতুন তথ্য ইনপুট করবো। এক্ষেত্রে আমরা `nextval` ফাংশনটি ব্যবহার করবো। এই ফাংশনটি "Student_id" সিকুয়েন্স থেকে পরবর্তী মান ধারন করবেঃ

```
INSERT INTO Student_details (Id, Roll_number,Student_name)
VALUES (Student_id.nextval, '১৩১','শাহরিয়ার হাসান');
Copy
```

উপরের SQL স্টেটমেন্টটি "Student_details" টেবিলে একটি নতুন তথ্য ইনপুট করবে। Student_id সিকুয়েন্স থেকে "Id" কলামে স্বয়ংক্রিয়ভাবেই ইউনিক মান যুক্ত হয়ে যাবে। "রোল নাম্বার(Roll_number)" কলামে "১৩১" এবং "শিক্ষার্থীর নাম(Student_name)" কলামে "শাহরিয়ার হাসান" মান যুক্ত করবে।

SQL কনস্ট্রেইন্ট

SQL কনস্ট্রেইন্ট(Constraints) একটি টেবিলের ডেটা কলামগুলির উপর নিয়ম আরোপ করতে ব্যবহৃত হয়। এটি একটি টেবিলে ইনপুটকৃত তথ্যের সীমা নির্ধারন করতে পারে। এটি ডেটাবেজ তথ্যের সঠিকতা এবং নির্ভরযোগ্যতা নিশ্চিত করে।

কনস্ট্রেইন্ট(constraint) কর্তৃক আরোপিত সীমা লংঘনকারী সকল কর্মকান্ডকে কনস্ট্রেইন্ট বাধা দেয়।

SQL এ `Constraints` সমূহঃ

- **`NOT NULL`** - একটি কলামে `NULL` ভ্যালু থাকতে পারবে না।
- **`UNIQUE`** - একটি কলামের প্রতিটি সারিতে ইউনিক ভ্যালু থাকবে।
- **`PRIMARY KEY`** - ইহা `NOT NULL` এবং `UNIQUE` এর সংমিশ্রণ।
- **`FOREIGN KEY`** - দুইটি টেবিলের মধ্যে সংযোগ সৃষ্টি করে।
- **`CHECK`**- একটি টেবিলের প্রতিটি সারিতে একটি শর্ত আরোপ করে।
- **`DEFAULT`** - একটি কলামের জন্য ডিফল্ট ভ্যালু নির্দিষ্ট করে।

টেবিল তৈরি করার সময়ে আমরা `Constraints` সেট করতে পারি। এছাড়া পূর্বের তৈরি টেবিলেও `Constraints` যোগ করতে পারি।

টেবিল তৈরির সময়ে SQL CONSTRAINT সিনটেক্স

```
CREATE TABLE name_of_table
(
```

```
name_of_column_1 data_type(size) name_of_constraint,
name_of_column_2 data_type(size) name_of_constraint,
....
);
Copy
```

SQL CONSTRAINT ডিলেট করা

```
ALTER TABLE name_of_table DROP CONSTRAINT name_of_constraint;
Copy
```

SQL Join

Join এর বাংলা অর্থ কোন কিছু একত্রিত করা। SQL -এ JOIN দুই বা ততোধিক টেবিলকে একত্রিত করে। একটি ডেটাবেজ দুই বা ততোধিক টেবিলের কলাম ফিল্ডের উপর ভিত্তিকরে যথাক্রমে দুই বা ততোধিক টেবিল থেকে সারি নিয়ে তাদের একত্রিত করার জন্য SQL JOIN `clause` ব্যবহৃত হয়।

ANSI স্ট্যান্ডার্ড অনুসারে SQL এ ৫ ধরনের JOIN রয়েছে

নিম্নে JOIN সমূহের তালিকা ও ব্যবহার তুলে ধরা হলোঃ

- **INNER JOIN** - উভয় টেবিলে অন্তত একটি কলামের মিল থাকলে সকল সারি রিটার্ন করে।
- **LEFT JOIN** - ডান টেবিলের মিলিত সারিসহ বাম টেবিলের সকল সারি রিটার্ন করে।
- **RIGHT JOIN** - বাম টেবিলের মিলিত সারিসহ ডান টেবিলের সকল সারি রিটার্ন করে।
- **FULL JOIN** -যেকোনো একটি টেবিলের সাথে মিল থাকলে উভয় টেবিলের সকল সারি রিটার্ন করে।
- **CROSS JOIN** - বাম পাশের মিলিত সারির একটি কলামের জন্য ডান পাশের মিলিত সারির প্রতিটি কলামকে রিটার্ন করে।

SQL এর সবচেয়ে সাধারণ JOIN হলোঃ SQL **INNER JOIN**।
join এর সাধারণ সর্ত পুরণ হলে SQL INNER JOIN একাধিক টেবিল থেকে সারি রির্টান করে।

নমুনা ডেটাবেজ

JOIN কীওয়ার্ডের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

নিচের অংশটি "Student_attendance" টেবিল থেকে নেওয়াঃ

রোল নাম্বার	উপস্থিতি	ভর্তির তারিখ
১০১	৮৯	০১-১১-২০১৫
১০২	৯১	০১-১১-২০১৫
১০৩	৮০	০১-১১-২০১৫
১০৪	৭৫	০২-১১-২০১৫
১০৫	৭৭	০২-১১-২০১৫

উপরের টেবিল দুটিতে লক্ষ্য করলে দেখবেন যে, "রোল নাম্বার(Roll_number)" কলামটি উভয় টেবিলে রয়েছে। "Student_details" টেবিলের "রোল নাম্বার(Roll_number)" কলামটি "Student_attendance" টেবিলের "রোল নাম্বার(Roll_number)" কলামকে নির্দেশ করে। "রোল নাম্বার(Roll_number)" কলামটি উভয় টেবিলের মধ্যে সম্পর্ক তৈরি করছে।

উদাহরণ

```
SELECT Student_attendance.Roll_number, Student_details.Student_name,
Student_attendance.Admission_date
FROM Student_attendance INNER JOIN Student_details
ON Student_attendance.Roll_number=Student_details.Roll_number;
Copy
```

উপরের উদাহরণটির ফলাফল নিম্নের ন্যায় দেখাবেঃ

রোল নাম্বার	শিক্ষার্থীর নাম	ভর্তির তারিখ
১০১	তামজীদ হাসান	০১-১১-২০১৫
১০২	মিনহাজুর রহমান	০১-১১-২০১৫
১০৩	মোঃ সবুজ হোসেন	০১-১১-২০১৫
১০৪	ইয়াসিন হোসেন	০২-১১-২০১৫

১০৫	ফরহাদ উদ্দিন	০২-১১-২০১৫

SQL INNER JOIN কীওয়ার্ড

Join এর সবচেয়ে গুরুত্বপূর্ণ এবং সর্বাধিক ব্যবহৃত ধরন হচ্ছে INNER JOIN। দুই বা ততোধিক টেবিলের কলামের ভ্যালু গুলোকে একত্রিত করার মাধ্যমে INNER JOIN একটি নতুন টেবিল তৈরি করে।
কুয়েরী করার সময় ON কীওয়ার্ড ের মাধ্যমে শর্ত জুড়ে দেওয়া হয়। যখন ঐ শর্ত বা শর্ত-সমূহ পূর্ণ হয় তখন উভয় টেবিলের তথ্য গুলো একত্রিত হয়ে একটি ফলাফল টেবিল তৈরি হয়।

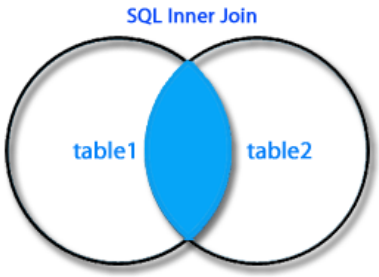
SQL INNER JOIN সিনটেক্স

```
SELECT name_of_column's
FROM first_table
INNER JOIN second_table
ON first_table.name_of_column=second_table.name_of_column;
Copy
```

অথবাঃ

```
SELECT name_of_column's
FROM first_table
JOIN second_table
ON first_table.name_of_column=second_table.name_of_column;
Copy
```

মনে রাখবেন, INNER JOIN এবং JOIN একই অর্থে ব্যবহৃত হয়।



নমুনা ডেটাবেজ

Collected By
Md. Jubayir Hossain

JOIN কীওয়ার্ডের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

নিচের অংশটি "Student_attendance" টেবিল থেকে নেওয়াঃ

রোল নাম্বার	উপস্থিতি	ভর্তির তারিখ
১০১	৮৯	০১-১১-২০১৫
১০২	৯১	০১-১১-২০১৫
১০৩	৮০	০১-১১-২০১৫
১০৪	৭৫	০২-১১-২০১৫
১০৫	৭৭	০২-১১-২০১৫

SQL INNER JOIN উদাহরণ

নিম্নের SQL স্টেটমেন্টটি সকল শিক্ষার্থীর নামের সাথে তাদের ভর্তির তারিখ ফেরত দিবেঃ

উদাহরণ

```
SELECT Student_attendance.Roll_number, Student_details.Student_name,
Student_attendance.Admission_date
FROM Student_attendance INNER JOIN Student_details
ON Student_attendance.Roll_number=Student_details.Roll_number;
Copy
```

বিঃদ্রঃ উভয় টেবিলের যেসকল কলামের তথ্য গুলো ম্যাচ করবে INNER JOIN শুধুমাত্র ঐ সকল সারি গুলোকে একত্রিত করে ফলাফল দেখাবে।

উপরের উদাহরণটির ফলাফল নিম্নের ন্যায় দেখাবেঃ

রোল নাম্বার	শিক্ষার্থীর নাম	ভর্তির তারিখ
১০১	তামজীদ হাসান	০১-১১-২০১৫
১০২	মিনহাজুর রহমান	০১-১১-২০১৫
১০৩	মোঃ সবুজ হোসেন	০১-১১-২০১৫
১০৪	ইয়াসিন হোসেন	০২-১১-২০১৫
১০৫	ফরহাদ উদ্দিন	০২-১১-২০১৫

SQL LEFT JOIN কীওয়ার্ড

SQL `LEFT JOIN` কীওয়ার্ডটি বাম টেবিলের(first_table) সকল সারিকে এবং ডান টেবিলের(second_table) শুধুমাত্র সদৃশ(matched) সারি গুলোকে একত্রিত করে **ফলাফল-টেবিলে** ফলাফল রিটার্ন করে। যদি সদৃশ কিছু খুঁজে না পায় তাহলে ডান টেবিল থেকে কোনো কিছু কুয়েরি/রিটার্ন করবে না।

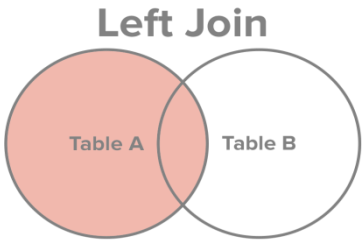
SQL LEFT JOIN সিনটেক্স

```
SELECT name_of_column's
FROM first_table
LEFT JOIN second_table
ON first_table.name_of_column=second_table.name_of_column;
Copy
```

অথবাঃ

```
SELECT name_of_column's
FROM first_table
LEFT OUTER JOIN second_table
ON first_table.name_of_column=second_table.name_of_column;
Copy
```

কিছু ডেটাবেজে `LEFT JOIN` কে `LEFT OUTER JOIN` বলা হয়ে থাকে।



নমুনা ডেটাবেজ

`LEFT JOIN` কীওয়ার্ডের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

নিচের অংশটি "Student_result" টেবিল থেকে নেওয়া হয়েছেঃ

রোল নাম্বার	ফলাফল
১০১	A+
১০২	A+
১০৩	A+
১০৪	A+
১০৫	A+

SQL LEFT JOIN উদাহরণ

নিম্নের SQL স্টেটমেন্টটি সকল শিক্ষার্থীর তথ্য রিটার্ন করবে এবং যদি তাদের পরীক্ষার ফলাফল থাকে তবে তাও রিটার্ন করবেঃ

উদাহরণ

```
SELECT Student_details.Student_name ,Student_details.Roll_number, Student_result.Result
FROM Student_details
LEFT JOIN Student_result
ON Student_details.Roll_number=Student_result.Roll_number
ORDER BY Student_details.Student_name;
Copy
```

বিঃদ্রঃ ডান টেবিলের মধ্যে সদৃশ(matched) কোনো কিছু খুঁজে না পেলেও `LEFT JOIN` কীওয়ার্ড টি বাম টেবিলের সকল সারি রিটার্ন করবে।

উপরের উদাহরণটির ফলাফল নিম্নের ন্যায় দেখাবেঃ

রোল নাম্বার	শিক্ষার্থীর নাম	ফলাফল
১২৩	আসমা আক্তার	A-
১০৪	ইয়াসিন হোসেন	A+
১২৮	উম্মে কুলসুম	B
১১৪	ওমর ফারুক	A
১০৯	ওয়াহিদুল ইসলাম	A

SQL RIGHT JOIN কীওয়ার্ড

SQL `RIGHT JOIN` কীওয়ার্ডটি ডান টেবিলের(second_table) সকল সারিকে এবং বাম টেবিলের(first_table) শুধুমাত্র সদৃশ(matched) সারি গুলোকে একত্রিত করে **ফলাফল-টেবিলে** ফলাফল রিটার্ন করে। যদি সদৃশ কিছু খুঁজে না পায় তাহলে বাম টেবিল থেকে কোনো কিছু কুয়েরি/রিটার্ন করবে না।

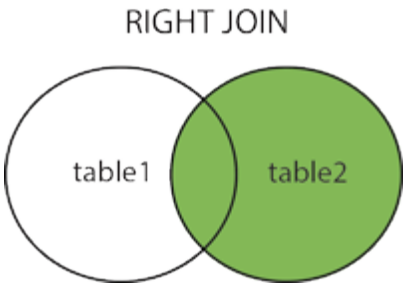
SQL RIGHT JOIN সিনট্যাক্স

```
SELECT name_of_column's
FROM first_table
RIGHT JOIN second_table
ON first_table.name_of_column=second_table.name_of_column;
Copy
```

অথবাঃ

```
SELECT name_of_column's
FROM first_table
RIGHT OUTER JOIN second_table
ON first_table.name_of_column=second_table.name_of_column;
Copy
```

কিছু ডাটাবেজে `RIGHT JOIN` কে `RIGHT OUTER JOIN` বলা হয়ে থাকে।



নমুনা ডেটাবেজ

`RIGHT JOIN` কীওয়ার্ডের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

নিচের অংশটি "Student_attendance" টেবিল থেকে নেওয়া হয়েছেঃ

রোল নাম্বার	উপস্থিতি	ভর্তির তারিখ
১০১	৮৯	০১-১১-২০১৫
১০২	৯১	০১-১১-২০১৫
১০৩	৮০	০১-১১-২০১৫
১০৪	৭৫	০২-১১-২০১৫
১০৫	৭৭	০২-১১-২০১৫

SQL RIGHT JOIN উদাহরণ

নিম্নের SQL স্টেটমেন্টটি সকল শিক্ষার্থীর তথ্য রিটার্ন করবে এবং যদি তাদের ভর্তির তারিখ দেওয়া থাকে তবে তাও রিটার্ন করবেঃ

উদাহরণ

```
SELECT Student_details.Roll_number, Student_details.Student_name,
Student_attendance.Admission_date
FROM Student_attendance
RIGHT JOIN Student_details
ON Student_details.Roll_number=Student_attendance.Roll_number
ORDER BY Student_attendance.Admission_date;
Copy
```

বিঃদ্রঃ বাম টেবিলের মধ্যে সদৃশ(matched) কোনো কিছু খুঁজে না পেলেও `RIGHT JOIN` কীওয়ার্ড টি ডান টেবিলের সকল সারি রিটার্ন করবে।

উপরের উদাহরণটির ফলাফল নিম্নের ন্যায় দেখাবেঃ

রোল নাম্বার	শিক্ষার্থীর নাম	ভর্তির তারিখ
১০১	তামজীদ হাসান	০১-১১-২০১৫
১০২	মিনহাজুর রহমান	০১-১১-২০১৫
১০৩	মোঃ সবুজ হোসেন	০১-১১-২০১৫
১০৪	ইয়াসিন হোসেন	০২-১১-২০১৫
১০৫	ফরহাদ উদ্দিন	০২-১১-২০১৫

SQL FULL JOIN কীওয়ার্ড

`FULL JOIN` কীওয়ার্ডটি বাম(first_table) এবং ডান(second_table) উভয় টেবিলের সকল সারি রিটার্ন করে।

সুতরাং `FULL JOIN` কীওয়ার্ডটি `LEFT JOIN` এবং `RIGHT JOIN` এর ফলাফল গুলো একত্রিত করে। এক্ষেত্রে যদি কোনো ফলাফল না পায় তাহলে `NULL` ভ্যালু রিটার্ন করে।

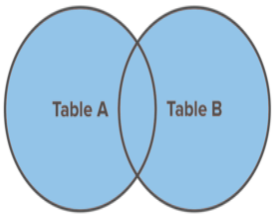
SQL FULL JOIN সিনট্যাক্স

```
SELECT name_of_column's
FROM first_table
FULL JOIN second_table
ON first_table.name_of_column=second_table.name_of_column;
Copy
```

অথবাঃ

```
SELECT name_of_column's
FROM first_table
FULL OUTER JOIN second_table
ON first_table.name_of_column=second_table.name_of_column;
Copy
```

কিছু ডেটাবেজে `FULL JOIN` কে `FULL OUTER JOIN` বলা হয়ে থাকে।



SQL FULL JOIN

নমুনা ডেটাবেজ

`FULL JOIN` কীওয়ার্ডের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

নিচের অংশটি "Student_attendance" টেবিল থেকে নেওয়া হয়েছেঃ

রোল নাম্বার	উপস্থিতি	ভর্তির তারিখ
১০১	৮৯	০১-১১-২০১৫
১০২	৯১	০১-১১-২০১৫
১০৩	৮০	০১-১১-২০১৫
১০৪	৭৫	০২-১১-২০১৫
১০৫	৭৭	০২-১১-২০১৫

SQL FULL JOIN উদাহরণ

নিম্নের SQL স্টেটমেন্টটি সকল শিক্ষার্থীর তথ্য গুলো একত্রিত করবেঃ

```
SELECT Student_details.Roll_number, Student_details.Student_name,
Student_attendance.Admission_date
FROM Student_details
FULL JOIN Student_attendance
ON Student_details.Roll_number=Student_attendance.Roll_number
ORDER BY Student_details.Student_name;
Copy
```

ফলাফলটি কিছুটা এমন দেখাবেঃ

রোল নাম্বার	শিক্ষার্থীর নাম	ভর্তির তারিখ
১০১	তামজীদ হাসান	০১-১১-২০১৫
১০২	মিনহাজুর রহমান	০১-১১-২০১৫
১০৩	মোঃ সবুজ হোসেন	০১-১১-২০১৫
১০৪	ইয়াসিন হোসেন	০১-১১-২০১৫
১০৫	ফরহাদ উদ্দিন	

বিঃদ্রঃ `FULL JOIN` কীওয়ার্ডটি বাম টেবিল(first_table) এবং ডান টেবিলের(second_table) সকল সারি রিটার্ন করবে। এক্ষেত্রে কোনো সদৃশ সারি না থাকলেও উভয় টেবিলের সকল সারি রিটার্ন করবে।

SQL UNION অপারেটর

SQL UNION অপারেটরটি দুই বা ততোধিক SELECT স্টেটমেন্টের ফলাফল একত্রে প্রকাশ করতে পারে।

SQL UNION অপারেটর

SQL UNION অপারেটরটি কোন ডুপ্লিকেট সারি ফেরত দেওয়া ছাড়াই দুই বা ততোধিক SELECT স্টেটমেন্টের ফলাফল একত্রিত করে।

UNION অপারেটর ব্যবহারের পূর্বশর্তঃ

- প্রতিটি SELECT স্টেটমেন্টে কলামের সংখ্যা অবশ্যই সমান থাকতে হবে।
- কলাম-সমূহের ডেটা টাইপ একই হতে হবে।
- SELECT স্টেটমেন্টের সকল কলাম-সমূহ একই ক্রমে(order) থাকতে হবে।

SQL UNION সিনট্যাক্স

```
SELECT name_of_column's FROM first_table
UNION
SELECT name_of_column's FROM second_table;
Copy
```

বিঃদ্রঃ UNION অপারেটরটি ডিফল্টভাবে একাধিক ভ্যালু শুধুমাত্র একবার সিলেক্ট করে। ডুপ্লিকেট(Duplicate) ভ্যালু পাওয়ার জন্য UNION এর সাথে ALL কিওয়ার্ড ব্যবহার করতে হবে।

SQL UNION ALL সিনট্যাক্স

```
SELECT name_of_column's FROM first_table
UNION ALL
SELECT name_of_column's FROM second_table;
Copy
```

UNION এ ব্যবহৃত প্রথম স্টেটমেন্টের কলামের নাম ফলাফল টেবিলের কলামের নাম এর সমান হয়। সুতরাং প্রথম স্টেটমেন্টের কলামের নাম-ই ফলাফল টেবিলের কলামের নাম হয়।

নমুনা ডেটাবেজ

UNION অপারেটরের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

নিচের অংশটি "Teacher_details" টেবিল থেকে নেওয়া হয়েছেঃ

রোল নাম্বার	শিক্ষার্থীর নাম	ঠিকানা
১০৪	ইয়াসিন হোসেন	
১০৫	ফরহাদ উদ্দিন	
১০৯	ওয়াহিদুল ইসলাম	

SQL UNION উদাহরণ

নিম্নের SQL UNION স্টেটমেন্টটি "Student_details" এবং "Teacher_details" টেবিল থেকে সকল "ঠিকানা(Address)" কলামকে সিলেক্ট করবে এবং শুধুমাত্র স্বতন্ত্র(Distinct) ভ্যালুগুলো নিয়ে আসবেঃ

উদাহরণ

```
SELECT Address FROM Student_details
UNION
SELECT Address FROM Teacher_details
ORDER BY Address;
Copy
```

বিঃদ্রঃ UNION অপারেটরটি দ্বারা "ঠিকানা(Address)" কলামের সকল ডেটা পাওয়া যাবে না। যদি এক বা একাধিক "শিক্ষার্থী" অথবা "শিক্ষক" এর শহর একই হয় তাহলে শহরটি একবার-ই দেখাবে। অপরপক্ষে UNION ALL ব্যবহার করলে সব গুলো শহরই একত্রে দেখাবে অর্থাৎ ডুপ্লিকেট ভ্যালু-সমূহও দেখাবে।

SQL UNION ALL উদাহরণ

নিম্নের SQL UNION ALL স্টেটমেন্টটি "Student_details" এবং "Teacher_details" টেবিল থেকে সকল স্বতন্ত্র "ঠিকানা"-সহ ডুপ্লিকেট ঠিকানাও নিয়ে আসবেঃ

উদাহরণ

```
SELECT Address FROM Student_details
UNION ALL
SELECT Address FROM Teacher_details
ORDER BY Address;
Copy
```

উপরের উদাহরণটির ফলাফল নিম্নের ন্যায় দেখাবেঃ

আইডি	ঠিকানা
১	ঢাকা
২	রাজশাহী
৩	চাঁদপুর
৪	বরিশাল
৫	সিলেট

SQL UNION ALL এর সাথে WHERE Clause এর ব্যবহার

নিম্নের SQL UNION ALL স্টেটমেন্টটি "Student_details" এবং "Teacher_details" টেবিল থেকে "**ঢাকা**" শহর বিশিষ্ট সকলের তথ্য(ডুপ্লিকেট ভ্যালুও) নিয়ে আসবেঃ

উদাহরণ

```
SELECT Student_name, Address FROM Student_details WHERE Address='ঢাকা'
UNION ALL
SELECT Teacher_name, Address FROM Teacher_details WHERE Address='ঢাকা'
ORDER BY Address;
Copy
```

উপরের উদাহরণটির ফলাফল নিম্নের ন্যায় দেখাবেঃ

আইডি	নাম	ঠিকানা
১	ওয়াহিদুল ইসলাম	ঢাকা
২	মারুফ হোসেন	ঢাকা
৩	ফারুক আলম	ঢাকা
৪	মোঃ সাইফুল ইসলাম	ঢাকা
৫	নাদিমা আক্তার	ঢাকা

SQL NULL ভ্যালু

NULL ভ্যালু দ্বারা যে সকল কলামে কোনো তথ্য থাকে না তাদেরকে বুঝায়। ডিফল্টভাবে একটি টেবিলের কলামে NULL ভ্যালু থাকতে পারে। এজন্য টেবিল তৈরির সময়েই কলামে NULL ভ্যালু ডিফাইন করে দিতে হবে।

SQL NULL ভ্যালু

একটি টেবিলের কোনো কলাম যদি ঐচ্ছিক(optional) হয় তাহলে আমরা ঐ কলামে কোনো ভ্যালু যোগ করা ছাড়াই নতুন রেকর্ড ইনসার্ট করতে অথবা পুরনো রেকর্ডকে আপডেট করতে পারি। এক্ষেত্রে ঐ ফিল্ডটি NULL ভ্যালু সংরক্ষন করবে।

অন্যান্য ভ্যালু হতে NULL ভ্যালুকে ভিন্নভাবে দেখা হয়। অজানা এবং প্রযোজ্যনহে এমন ভ্যালুর জন্য NULL ব্যবহার করা হয়।

বিঃদ্রঃ আপনি NULL এবং শূণ্য(0) এর মধ্যে তুলনা করতে পারবেন না কারণ তারা সমান এবং সমজাতীয় নয়।

SQL এ NULL ভ্যালুর ব্যবহার

নিচের "Student_details" টেবিলে লক্ষ্য করুনঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

ধরুন "Student_details" টেবিলের "ঠিকানা(Address)" কলামটি ঐচ্ছিক। আমরা যদি "ঠিকানা(Address)" কলামে কোন ভ্যালু ছাড়াই একটি রেকর্ড তৈরি করি তাহলে "ঠিকানা(Address)" কলামের ভ্যালু NULL হবে।

আপনি NULL ভ্যালুকে কম্প্যারিজন(comparison) অপারেটর(=, <, <>) এর মাধ্যমে যাচাই করতে পারবেন না।

বরং এর পরিবর্তে আপনি IS NULL এবং IS NOT NULL অপারেটর ব্যবহার করতে পারেন।

SQL IS NULL অপারেটর

আমরা "ঠিকানা(Address)" কলামের NULL ভ্যালুযুক্ত রেকর্ডকে IS NULL অপারেটর ব্যবহার করে সিলেক্ট করতে পারিঃ

```
SELECT Roll_number, Student_name, Address
FROM Student_details
WHERE Address IS NULL;
Copy
```

ফলাফলঃ

রোল নাম্বার	শিক্ষার্থীর নাম	ঠিকানা
১০৪	ইয়াসিন হোসেন	
১০৫	ফরহাদ উদ্দিন	
১০৯	ওয়াহিদুল ইসলাম	

পরামর্শঃ `NULL` ভ্যালু খুঁজে বের করার জন্য সব সময় `IS NULL` অপারেটর ব্যবহার করুন।

SQL IS NOT NULL অপারেটর

"ঠিকানা(Address)" কলামে `NULL` ভ্যালু নেই এমন রেকর্ডকে সিলেক্ট করতে আমরা `IS NOT NULL` অপারেটর ব্যবহার করতে পারিঃ

```
SELECT Roll_number, Student_name, Address
FROM Student_details
WHERE Address IS NOT NULL;
Copy
```

ফলাফলঃ

রোল নাম্বার	শিক্ষার্থীর নাম	ঠিকানা
১০১	তামজীদ হাসান	চাঁদপুর
১০২	মিনহাজুর রহমান	চাঁদপুর
১০৩	মোঃ সবুজ হোসেন	চাঁদপুর

পরবর্তীতে আমরা `ISNULL()`, `NVL()`, `IFNULL()` এবং `COALESCE()` ফাংশন সম্পর্কে জানবো।

SQL NULL ফাংশন

SQL ISNULL(), NVL(), IFNULL() এবং COALESCE() ফাংশন

নিম্নের "Student_result" টেবিলটি লক্ষ্য করুনঃ

আইডি নং	রোল নাম্বার	লিখিত নাম্বার	এমসিকিউ নাম্বার	ফলাফল
১	১০১	৫১	৩৬	A+
২	১০২	৫২	৩৫	A+
৩	১০৩	৫৪	৩০	A+
৪	১০৪	৫০	৩১	A+
৫	১০৫	৪৯	৩৩	A+

ধরুন "এমসিকিউ নাম্বার(Mcq_number)" কলামটি ঐচ্ছিক এবং ইহা `Null` ভ্যালু ধারণ করতে পারে।

নিম্নের উদাহরণটি লক্ষ করুনঃ

```
SELECT Roll_number, (Written_number+Mcq_number)
FROM Student_result;
Copy
```

উপরের উদাহরণে যদি "এমসিকিউ নাম্বার(Mcq_number)" কলামের ভ্যালু `Null` হয় তাহলে ফলাফলও `Null` হবে।

`Null` এর ভ্যালু নির্ধারণের জন্য মাইক্রোসফট `ISNULL()` ফাংশন ব্যবহার করে।

`NVL()`, `IFNULL()` এবং `COALESCE()` ফাংশন গুলোও একই কাজ করে।

`Null` ফাংশন ব্যবহারের উদ্দেশ্য হলো আমরা `Null` ভ্যালুর পরিবর্তে 0(শূন্য) পেতে চাই।

`ISNULL()` ফাংশন ব্যবহার করলে "এমসিকিউ নাম্বার(Mcq_number)" কলামের কোন মান `Null` হলেও হিসাব-নিকাশে কোন সমস্যা হবে না, কারন প্রতিটি `Null` ভ্যালুর জন্য `ISNULL()` ফাংশনটি একটি 0(শূন্য) রিটার্ন করবেঃ

MS Access এর জন্য

```
SELECT Roll_number, (Written_number+IIF(ISNULL(Mcq_number),0,Mcq_number))
FROM Student_result;
Copy
```

SQL Server এর জন্য

```
SELECT Roll_number, (Written_number+ISNULL(Mcq_number,0))
FROM Student_result;
Copy
```

ওরাকলে কোনো `ISNULL()` ফাংশন নেই। ওরাকলে আমরা `ISNULL()` এর পরিবর্তে `NVL()` ফাংশনটি ব্যবহার করবোঃ

Oracle এর জন্য

```
SELECT Roll_number, (Written_number+NVL(Mcq_number,0))
FROM Student_result;
Copy
```

MySQL এ `ISNULL()` ফাংশন আছে। তবুও ইহা মাইক্রোসফট `ISNULL()` ফাংশনের চেয়ে একটু ভিন্ন ভাবে কাজ করে।

MySQL এ আমরা `IFNULL()` ফাংশনটি ব্যবহার করবোঃ

MySQL এর জন্য

```
SELECT Roll_number, (Written_number+IFNULL(Mcq_number,0))
FROM Student_result;
Copy
```

অথবা আমরা `COALESCE()` ফাংশনটি ব্যবহার করতে পারিঃ

```
SELECT Roll_number, (Written_number+COALESCE(Mcq_number,0))
FROM Student_result;
```

SQL Alias

অস্থায়ীভাবে একটি টেবিল বা একটি কলামের নাম পরিবর্তন করতে SQL `alias` ব্যবহার করা হয়।

SQL Alias

SQL `alias` এর মাধ্যমে ডাটাবেজ টেবিল অথবা টেবিল কলামের জন্য একটি অস্থায়ী নাম দেওয়া হয়। `Alias` ব্যবহার করলে ডেটাবেজের মূল টেবিল বা কলামের নামের কোন পরিবর্তন হয় না।

সাধারনত কলামের নাম-সমূহকে অধিক পাঠযোগ্য করে তোলার জন্য `alias` তৈরি করা হয়।

কলামের জন্য SQL Alias সিনট্যাক্স

```
SELECT name_of_column AS name_of_alias
FROM name_of_table;
Copy
```

টেবিলের জন্য SQL Alias সিনট্যাক্স

```
SELECT name_of_column's
FROM name_of_table AS name_of_alias;
```

Alias কখন ব্যবহার করবেন?

- যখন কুয়েরির মধ্যে এক বা একধিক টেবিল জড়িত থাকে।
- যখন কুয়েরির মধ্যে ফাংশন ব্যবহার করা হয়।
- যখন কলামের নাম বড় অথবা পাঠযোগ্য না হয়।
- যখন এক বা একাধিক কলামের সন্নিবেশ ঘটানো হয়।

নমুনা ডেটাবেজ

AS কীওয়ার্ড এর ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

নিচের অংশটি "Student_attendance" টেবিল থেকে নেওয়া হয়েছেঃ

রোল নাম্বার	উপস্থিতি	ভর্তির তারিখ
১০১	৮৯%	০১-১১-২০১৫
১০২	৯১%	০১-১১-২০১৫
১০৩	৮০%	০১-১১-২০১৫
১০৪	৭৫%	০২-১১-২০১৫
১০৫	৭৭%	০২-১১-২০১৫

কলামের জন্য Alias এর উদাহরণ

নিচের SQL স্টেটমেন্টটিতে আমরা দুইটি `alias` ব্যবহার করবো। একটি "Student_name" কলামের জন্য এবং অন্যটি "Address" কলামের জন্য।

বিঃদ্রঃ যদি কলাম নামে স্পেস থাকে তাহলে ডাবল উদ্ধৃতি("") অথবা স্কোয়ার ব্যাকেট ব্যবহার করতে হবেঃ

উদাহরণ

```
SELECT Student_name AS "শিক্ষার্থীর নাম", Address AS "ঠিকানা"
FROM Student_details;
```

ফলাফলঃ

শিক্ষার্থীর নাম	ঠিকানা
তামজীদ হাসান	চাঁদপুর
মিনহাজুর রহমান	চাঁদপুর
মোঃ সবুজ হোসেন	চাঁদপুর
ইয়াসিন হোসেন	চাঁদপুর
ফরহাদ উদ্দিন	চাঁদপুর

নিচের SQL স্টেটমেন্টটিতে আমরা Institute এবং Address কলাম দুটি একত্রিত করে "প্রতিষ্ঠানের ঠিকানা(Institute_address)" নামে একটি নতুন `alias` তৈরি করবোঃ

উদাহরণ

```
SELECT Student_name, Institute+', '+Address AS Institute_address
FROM Student_details;
```

ফলাফলঃ

শিক্ষার্থীর নাম	প্রতিষ্ঠানের ঠিকানা
তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়, চাঁদপুর
মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়, চাঁদপুর
মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়, চাঁদপুর
ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়, চাঁদপুর
ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়, চাঁদপুর

বিঃদ্রঃ উপরের SQL স্টেটমেন্টটি MySQL এ ঠিকমত কাজ করানোর জন্য নিম্নের কোড অনুসরন করুনঃ

```
SELECT Student_name, CONCAT(Institute,', ',Address)
AS Institute_address
FROM Student_details;
Copy
```

টেবিলের জন্য Alias এর উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "Student_details" এবং "Student_attendance" টেবিল থেকে "রোল নাম্বার, শিক্ষার্থির নাম" এবং "ভর্তির তারিখ" কলামের রেকর্ড সিলেক্ট করেবে। কিন্তু শুধুমাত্র "শিক্ষার্থির নাম(Student_name)" কলামের "তামজীদ হাসান" এর রেকর্ড গুলো দেখাবেঃ

উদাহরণ

```
SELECT Sd.Roll_number, Sd.Student_name, Sa.Admission_date
FROM Student_details AS Sd, Student_attendance AS Sa
WHERE Sd.Student_name = "তামজীদ হাসান" AND Sd.Roll_number=Sa.Roll_number;
Copy
```

ফলাফলঃ

রোল নাম্বার	শিক্ষার্থীর নাম	ভর্তির তারিখ
১০১	তামজীদ হাসান	০১-১১-২০১৫

`alias` ব্যতিত একই SQL স্টেটমেন্টঃ

উদাহরণ

```
SELECT Student_details.Roll_number, Student_details.Student_name,
Student_attendance.Admission_date
FROM Student_details, Student_attendance
WHERE Student_details.Student_name = "তামজীদ হাসান" AND
Student_details.Roll_number=Student_attendance.Roll_number;
Copy
```

ফলাফলঃ

রোল নাম্বার	শিক্ষার্থীর নাম	ভর্তির তারিখ
১০১	তামজীদ হাসান	০১-১১-২০১৫

SQL CREATE INDEX স্টেটমেন্ট

`CREATE INDEX` স্টেটমেন্ট ব্যবহার করে টেবিলের মধ্যে ইনডেক্স তৈরি করা হয়। একটি টেবিলের সম্পূর্ন তথ্য না পড়েই, ইনডেক্সের সাহায্য ডাটাবেজ থেকে দ্রুত তথ্য খুঁজে পাওয়া যায়।

ইনডেক্স(INDEX)

একটি ডাটাবেজ টেবিল থেকে দ্রুত এবং দক্ষতার সাথে তথ্য খুঁজে বের করার জন্য ইনডেক্স তৈরি করা হয়। ইউজাররা ইনডেক্স দেখতে পায় না, এগুলো শুধুমাত্র দ্রুত তথ্য কুয়েরি/খুঁজে বের করার জন্য ব্যবহার করা হয়।

বিঃদ্রঃ সাধারন টেবিলের তথ্য আপডেট করতে যে সময় লাগে ইনডেক্স যুক্ত টেবিলের তথ্য আপডেট করতে তার চেয়ে বেশী সময় লাগে। সুতরাং সচারচর সার্স করতে হবে এমন কলামের জন্য ইনডেক্স তৈরি করুন।

SQL CREATE INDEX সিনট্যাক্স

```
CREATE INDEX name_of_index
ON name_of_table (name_of_column);
Copy
```

ইনডেক্স টেবিলে ডুপ্লিকেট ভ্যালু গ্রহণযোগ্য।

SQL CREATE UNIQUE INDEX সিনট্যাক্স

```
CREATE UNIQUE INDEX name_of_index
ON name_of_table (name_of_column);
Copy
```

ইউনিক ইনডেক্স টেবিলে ডুপ্লিকেট ভ্যালু গ্রহণযোগ্য নহে।

বিঃদ্রঃ ইনডেক্স তৈরির সিনট্যাক্স বিভিন্ন ডেটাবেজে বিভিন্ন রকম হয়। সুতরাং ইনডেক্স তৈরির পূর্বে আপনার ডেটাবেজের জন্য ইনডেক্স তৈরির সিনট্যাক্সটি দেখে নিন।

নমুনা ডেটাবেজ

`CREATE INDEX` স্টেটমেন্টের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

CREATE INDEX এর উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিলের "শিক্ষার্থীর নাম(Student_name)" কলামে "Sn_Index" নামে একটি ইনডেক্স তৈরি করবেঃ

```
CREATE INDEX Sn_Index
ON Student_details (Student_name);
Copy
```

আপনি যদি দুইটি কলামের জন্য একটি ইনডেক্স তৈরি করতে চান তাহলে নিম্নের SQL স্টেটমেন্টটি দেখুনঃ

```
CREATE INDEX Sa_Index
ON Student_details (Institute, Address);
```

SQL ALTER TABLE স্টেটমেন্ট

একটি বিদ্যমান টেবিলে নতুন কলাম যোগ করতে, কলাম ডিলেট করতে অথবা কোন কলাম পরিবর্তন করতে `ALTER TABLE` স্টেটমেন্ট ব্যবহার করা হয়।

SQL ALTER TABLE সিনট্যাক্স

একটি টেবিলে নতুন কলাম যোগ করতে নিম্নের সিনট্যাক্সটি ব্যবহার করুনঃ

```
ALTER TABLE name_of_table
ADD name_of_column datatype;
Copy
```

টেবিলে থেকে কলাম ডিলেট করতে নিম্নের সিনট্যাক্সটি ব্যবহার করুনঃ

```
ALTER TABLE name_of_table
DROP name_of_column;
Copy
```

টেবিলে কলামের ডেটা টাইপ পরিবর্তন করতে নিম্নের সিনট্যাক্সটি ব্যবহার করুনঃ

MySQL/Oracle(10G এর আগের ভার্সন) এর জন্যঃ

```
ALTER TABLE name_of_table
MODIFY COLUMN name_of_column datatype;
Copy
```

Oracle(10G এবং এর পরবর্তী ভার্সন) এর জন্যঃ

```
ALTER TABLE name_of_table
MODIFY name_of_column datatype;
Copy
```

SQL Server/MS Access এর জন্যঃ

```
ALTER TABLE name_of_table
ALTER COLUMN name_of_column datatype;
Copy
```

SQL ALTER TABLE এর উদাহরণ

নিম্নের "Student_details" টেবিলটি লক্ষ্য করুনঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

এখন আমরা "Student_details" টেবিলে "জন্ম তারিখ(Birthday)" নামে একটি কলাম যোগ করবো।

এর জন্য আমরা নিম্নবর্তী SQL স্টেটমেন্টটি ব্যবহার করবোঃ

```
ALTER TABLE Student_details
ADD Birthday date;
Copy
```

লক্ষ্য করুন "জন্ম তারিখ(Birthday)" কলামটির ডেটা টাইপ হলো `date` অর্থাৎ এটি তারিখ জমা রাখবে।

এখন "Student_details" টেবিলেটি নিম্নের ন্যায় দেখাবেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা	জন্ম তারিখ
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর	
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর	
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর	
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর	
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর	

ডেটা টাইপ পরিবর্তনের উদাহরণ

এখন আমরা "Student_details" টেবিলের "জন্ম তারিখ(Birthday)" কলামের ডেটা টাইপ পরিবর্তন করবোঃ

```
ALTER TABLE Student_details
ALTER COLUMN Birthday year;
Copy
```

লক্ষ্য করুন "জন্ম তারিখ(Birthday)" কলামটির ডেটা টাইপ এখন `year` অর্থাৎ এটি শুধুমাত্র দুই/চার ডিজিটের ফরম্যাটে বছর জমা রাখবে।

DROP COLUMN এর উদাহরণ

এখন আমরা "Student_details" টেবিলের "জন্ম তারিখ(Birthday)" কলামটিকে ডিলেট করবোঃ

```
ALTER TABLE Student_details
DROP COLUMN Birthday;
Copy
```

এখন "Student_details" টেবিলেটি নিম্নের ন্যায় দেখাবেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

SQL VIEW স্টেটমেন্ট

`VIEW` হল একটি ভার্চুয়াল টেবিল। এই অধ্যায়ে আমরা একটি `VIEW` তৈরী, আপডেট এবং ডিলেট করা শিখবো।

SQL CREATE VIEW স্টেটমেন্ট

SQL এ `VIEW` একটি ভার্চুয়াল টেবিল যা SQL স্টেটমেন্টের রেজাল্ট-সেট এর উপর ভিত্তিকরে গঠিত হয়।

একটি বাস্তব টেবিলের ন্যায় `VIEW` টেবিলেও কলাম এবং সারি থাকে। ডেটাবেজের অন্তর্ভূক্ত এক বা একধিক টেবিলের কলাম `VIEW` টেবিলের কলাম হয়।

তথ্য দেখা এবং প্রদর্শনীর জন্য আপনি VIEW টেবিলে **SQL ফাংশন**, WHERE এবং JOIN স্টেটমেন্ট ব্যবহার করতে পাবেন।

SQL CREATE VIEW সিনট্যাক্স

```
CREATE VIEW name_of_view AS
SELECT name_of_column's
FROM name_of_table
WHERE condition;
Copy
```

বিঃদ্রঃ VIEW সর্বদাই আপনাকে আপডেট তথ্য দেখাবে! যখন কোনো ইউজার VIEW কুয়েরি করে তখন ডাটাবেজ ইঞ্জিন SQL VIEW স্টেটমেন্ট ব্যবহার করে পুনরায় তথ্য তৈরি করে।

SQL CREATE VIEW স্টেটমেন্টের উদাহরণ

আপনি যদি আমাদের নমুনা ডেটাবেজটি দেখে থাকেন তাহলে নিশ্চয়ই দেখবেন যে, ডিফল্ট ভাবে ইহার বিভিন্ন ধরনের ভিউ আছে।

আমরা "Student_details" টেবিল থেকে "বর্তমান শিক্ষার্থীর তালিকা(Current Student List)" এর জন্য একটি VIEW তৈরী করবো। যেখানে বর্তমানে প্রাপ্ত সকল শিক্ষার্থীর তথ্য থাকবে। এর জন্য নিম্নের SQL স্টেটমেন্টটি ব্যবহার করুনঃ

```
CREATE VIEW [Current Student List] AS
SELECT Roll_number, Student_name
FROM Student_details
WHERE Discontinued=No;
Copy
```

উপরের VIEW কে আমরা নিম্নের ন্যায়ও কুয়েরি করতে পারিঃ

```
SELECT * FROM [Current Student List]
Copy
```

নিম্নের SQl স্টেটমেন্ট ব্যবহার করে অন্য একটি VIEW এ আমরা আমাদের নমুনা ডেটাবেজের "Student_details" টেবিলের সকল শিক্ষার্থীকে তাদের রোল নাম্বারের ক্রমানুসারে দেখাবোঃ

```
CREATE VIEW [Student Serialized by Roll] AS
SELECT Roll_number, Student_name
FROM Student_details
WHERE Roll_number>(SELECT AVG(Roll_number) FROM Student_details);
Copy
```

উপরের VIEW কে আমরা নিম্নের ন্যায়ও কুয়েরি করতে পারিঃ

```
SELECT * FROM [Student Serialized by Roll]
Copy
```

আমাদের নমুনা ডেটাবেজের অন্য একটি VIEW -এ আমরা দেখবো "Student_details" টেবিলের সকল শিক্ষার্থীদের মধ্যে কারা ২০১৫ সালে ভর্তি হয়েছিল। এই VIEW এ যে তথ্যগুলো দেখানো হয়েছে তা অন্য একটি VIEW "Student Details 2015" থেকে নেওয়া হয়েছেঃ

```
CREATE VIEW [Admitted Student 2015] AS
SELECT Roll_number, Student_name
FROM [Student Details 2015]
Group BY Student_name;
Copy
```

উপরের VIEW কে আমরা নিম্নের ন্যায়ও কুয়েরি করতে পার

```
SELECT * FROM [Student Details 2015]
Copy
```

কুয়েরিতে আমরা শর্তও যোগ করতে পারিঃ

```
SELECT * FROM [Student Details 2015]
WHERE Student_name='তামজীদ হাসান'
Copy
```

SQL VIEW আপডেট

নিম্নবর্তী সিনট্যাক্সটি ব্যবহার করে VIEW আপডেট করতে পারিঃ

SQL CREATE অথবা REPLACE VIEW সিনট্যাক্স

```
CREATE OR REPLACE VIEW name_of_view AS
SELECT name_of_column's
FROM name_of_table
WHERE condition
Copy
```

এখন আমরা "বর্তমান শিক্ষার্থীর তালিকায়(Current Student List)" "জন্মদিন(Birthday)" কলামটি যোগ করবো। আমরা নিম্নবর্তী সিনট্যাক্স দ্বারা VIEW আপডেট করবোঃ

```
CREATE OR REPLACE VIEW [Current Student List] AS
SELECT Roll_number, Student_name, Birthday
FROM Student_details
```

```
WHERE Discontinued=No;
Copy
```

SQL ডিলেট VIEW

আপনি `DROP VIEW` কমান্ডের মাধ্যমে একটি `VIEW` ডিলেট করতে পারেন।

SQL DROP VIEW সিনট্যাক্স

```
DROP VIEW name_of_view
```

SQL HAVING Clause

SQL HAVING Clause

`Aggregate` ফাংশন এর সাথে `WHERE` কিওয়ার্ড ব্যবহার করা যেত না বলে SQL এ `HAVING clause` যোগ করা হয়েছিল।

SQL HAVING সিনট্যাক্স

```
SELECT name_of_column, aggregate_function(name_of_column)
FROM name_OF_table
WHERE name_of_column operator value
GROUP BY name_of_column
HAVING aggregate_function(name_of_column) operator value;
Copy
```

নমুনা ডেটাবেজ

`HAVING Clause` এর ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

নিচের অংশটি "Student_attendance" টেবিল থেকে নেওয়াঃ

আইডি নং	রোল নাম্বার	উপস্থিতি	ভর্তির তারিখ
১	১০১	৮৯	০১-১১-২০১৫
২	১০২	৯১	০১-১১-২০১৫
৩	১০৩	৮০	০১-১১-২০১৫
৪	১০৪	৭৫	০২-১১-২০১৫
৫	১০৫	৭৭	০২-১১-২০১৫

SQL HAVING Clause এর উদাহরণ

নিচের SQL স্টেটমেন্টটি যে সকল শিক্ষার্থী ৭৫ শতাংশ উপস্থিত ছিল তাদেরকে সিলেক্ট করবে।

উদাহরণ

```
SELECT Student_details.Student_name, Student_attendance.Roll_number,
COUNT(Student_attendance.Attendance) AS Attendance
FROM (Student_attendance
INNER JOIN Student_details
ON Student_attendance.Roll_number=Student_details.Roll_number)
GROUP BY Student_name
HAVING COUNT(Student_attendance.Attendance) > 75;
Copy
```

ফলাফল

রোল নাম্বার	শিক্ষার্থীর নাম	শতকরা উপস্থিতি
১০১	তামজীদ হাসান	৮৯%
১০২	মিনহাজুর রহমান	৯১%
১০৩	মোঃ সবুজ হোসেন	৮০%
১০৪	ইয়াসিন হোসেন	৮৭%
১০৫	ফরহাদ উদ্দিন	৮৫%

এখন আমরা খুজে বের করবো, শিক্ষার্থী "তামজীদ হসান" অথবা "মিনহাজুর রহমান" শতকরা ৯০ শতাংশ উপস্থিত ছিল কি না।

উদাহরণ

```
SELECT Student_details.Student_name, Student_attendance.Roll_number,
COUNT(Student_attendance.Attendance) AS Attendance
FROM (Student_attendance
```

```
INNER JOIN Student_details
ON Student_attendance.Roll_number=Student_details.Roll_number)
WHERE Student_name="তামজীদ হসান" OR Student_name="মিনহাজুর রহমান"
GROUP BY Student_name
HAVING COUNT(Student_attendance.Attendance) > 90;
Copy
```

ফলাফল

রোল নাম্বার	শিক্ষার্থীর নাম	শতকরা উপস্থিতি
১০২	মিনহাজুর রহমান	৯১%

SQL ওয়াইল্ডকার্ড

একটি স্ট্রিং এর মধ্যে যেকোন ক্যারেক্টার এর বিকল্প হিসাবে **ওয়াইল্ডকার্ড(wildcard)** ক্যারেক্টার ব্যবহার করা হয়।

SQL ওয়াইল্ডকার্ড ক্যারেক্টার

SQL এ LIKE অপারেটরের সাথে ওয়াইল্ডকার্ড ক্যারেক্টার ব্যবহার করা হয়। SQL ওয়াইল্ডকার্ড ব্যবহার করে টেবিলের মধ্য থেকে ডেটা সার্চ করা হয়।

নিম্নে SQL ওয়াইল্ডকার্ড গুলো বর্ণনা করা হলঃ

ওয়াইল্ডকার্ড	বর্ণনা
%	শূন্য বা অধিক ক্যারেক্টার খুঁজে করে।
_	একটি একক ক্যারেক্টার খুঁজে বের করে।
[charlist]	এক সেট অথবা নির্দিষ্ট ব্যবধী হতে ক্যারেক্টার খুঁজে বের করে।
[^charlist] অথবা [!charlist]	বন্ধনীতে উল্লেখিত ক্যারেক্টার ব্যতিত বাকি সকল ক্যারেক্টার খুঁজে বের করে।

নমুনা ডেটাবেজ

ওয়াইল্ডকার্ড এর ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

SQL % ওয়াইল্ডকার্ডের ব্যবহার

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিল থেকে সকল তথ্য সিলেক্ট করবে, কিন্তু "ঠিকানা(Address)" কলামে অবস্থিত যে সকল শব্দ "ঢা" দিয়ে শুরু হয়েছে শুধুমাত্র তাদের দেখাবেঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE Address LIKE 'ঢা%';
Copy
```

ফলাফলঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
৯	১০৯	ওয়াহিদুল ইসলাম	জাতীয় বিশ্ববিদ্যালয়	ঢাকা
১১	১১১	সৌরভ বনিক	জাতীয় বিশ্ববিদ্যালয়	ঢাকা
১৬	১১৬	মারুফ হোসেন	জাতীয় বিশ্ববিদ্যালয়	ঢাকা
২০	১২০	দেলোয়ার হোসেন	জাতীয় বিশ্ববিদ্যালয়	ঢাকা
২২	১২২	ফারুক আলম	জাতীয় বিশ্ববিদ্যালয়	ঢাকা

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিল থেকে সকল তথ্য সিলেক্ট করবে, কিন্তু "ঠিকানা(Address)" কলামে অবস্থিত যে সকল শব্দে "দপু" প্যাটার্ন থাকবে শুধুমাত্র তাদের দেখাবেঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE Address LIKE '%দপু%';
Copy
```

ফলাফলঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা

১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

SQL _ ওয়াইল্ডকার্ডের ব্যবহার

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিল থেকে সকল তথ্য সিলেক্ট করবে, কিন্তু "ঠিকানা(Address)" কলামে অবস্থিত যে সকল শব্দ যেকোন অক্ষর দিয়ে শুরু হবে এবং এর পরে "জশাহী" প্যাটার্ন থাকবে শুধুমাত্র তাদের দেখাবেঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE Address LIKE '_ _জশাহী';
Copy
```

"ঠিকানা(Address)" কলামে অবস্থিত যে সকল শব্দ যেকোন অক্ষর দিয়ে শুরু হবে এর পরে "া" এবং এর পরে যেকোনো সংখ্যা এবং এর পরে "শাহী" প্যাটার্ন থাকবে শুধুমাত্র তাদের দেখাবেঃ

```
SELECT * FROM Student_details
WHERE Address LIKE '_ া _শাহী';
Copy
```

ফলাফলঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
৭	১০৭	মোঃ ফয়সাল ইসলাম	জাতীয় বিশ্ববিদ্যালয়	রাজশাহী
১৪	১১৪	ওমর ফারুক	জাতীয় বিশ্ববিদ্যালয়	রাজশাহী
১৯	১১৯	মোঃ মমিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	রাজশাহী
৩০	১৩০	হেদায়েত উল্লাহ	জাতীয় বিশ্ববিদ্যালয়	রাজশাহী

SQL [charlist] ওয়াইল্ডকার্ডের ব্যবহার

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিল থেকে সকল তথ্য সিলেক্ট করেবে, কিন্তু "ঠিকানা(Address)" কলামে অবস্থিত শব্দ গুলোর মধ্যে যাদের প্রথম অক্ষর "ঢ" অথবা "র" দিয়ে শুরু হয়ছে শুধুমাত্র তাদের দেখাবেঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE Address LIKE '[ঢর]%';
Copy
```

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিল থেকে সকল তথ্য সিলেক্ট করবে, কিন্তু "ঠিকানা(Address)" কলামে অবস্থিত শব্দগুলোর মধ্যে যাদের প্রথম অক্ষর "ক" থেকে "চ" এর মধ্যে যেকোন একটি দিয়ে শুরু হয়েছে শুধুমাত্র তাদের দেখাবেঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE Address LIKE '[ক-চ]%';
Copy
```

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিল থেকে সকল তথ্য সিলেক্ট করবে, কিন্তু "ঠিকানা(Address)" কলামে অবস্থিত শব্দ গুলোর মধ্যে যাদের প্রথম অক্ষর "ঢ" অথবা "র" দিয়ে শুরু হয়েছে শুধুমাত্র তাদের দেখাবে নাঃ

উদাহরণ

```
SELECT * FROM Student_details
WHERE Address LIKE '[!ঢর]%';
Copy
```

অথবা

উদাহরণ

```
SELECT * FROM Student_details
WHERE Address NOT LIKE '[ঢর]%';
```

বিঃদ্রঃ উপরের সকল উদাহণের ক্ষেত্রে বাংলা ক্যারেক্টার সঠিকভাবে কাজ নাও করতে পারে।

SQL Date ফাংশন

তারিখ নিয়ে কাজ করার সময় সবচেয়ে গুরত্বপূর্ণ কাজ হলো আপনি তারিখের যে ফরম্যাট নির্বাচন করেছেন তা যেন আপনার ডেটাবেজ কলামের `date` ফরম্যাটের সাথে মিলে। এটা নাহলে আপনি ডেটাবেজে সঠিক ভাবে `date` ইনপুট করতে পারবেন না।

আপনার `date` কলামে শুধুমাত্র তারিখ থাকলে কুয়েরি করতে অনেক সহজ হবে। কিন্তু যদি আপনার `date` কলামে তারিখের পাশাপাশি সময়ও থাকে তাহলে কুয়েরি করা একটু কঠিন হয়ে উঠে।

নিম্নে MySQL এবং SQL Server এর কিছু গুরুত্বপূর্ণ বিল্ট-ইন `date` ফাংশন এবং তাদের ব্যবহার বর্ণনা করা হলোঃ

MySQL Date ফাংশন

নিম্নের তালিকায় MySQL এর কয়েকটি গুরুত্বপূর্ণ বিল্ট-ইন ফাংশন দেওয়া হলঃ

ফাংশন	বর্ণনা
`NOW()`	বর্তমান `date` এবং `time` রিটার্ন করবে।
`CURDATE()`	বর্তমান `date` রিটার্ন করবে।
`CURTIME()`	বর্তমান `time` রিটার্ন করবে।
`DATE()`	`date` অথবা `date/time` এক্সপ্রেশন থেকে শুধুমাত্র `date` অংশটি নিবে।
`EXTRACT()`	`date` অথবা `time` যেকোন একটি অংশ রিটার্ন করবে।
`DATE_ADD()`	`date` এ একটি নির্দিষ্ট সময় ব্যবধান যোগ করবে।
`DATE_SUB()`	`date` থেকে একটি নির্দিষ্ট সময় ব্যবধান বিয়োগ করবে।
`DATEDIFF()`	দুইটি `date` এর পার্থক্য রিটার্ন করবে।
`DATE_FORMAT()`	`date` অথবা `time` কে দেখানোর জন্য বিভিন্ন ফরম্যাট নির্ধারন করবে।

SQL Server Date ফাংশন

নিম্নের লিস্টে SQL Server এর কয়েকটি গুরুত্বপূর্ণ বিল্ট-ইন ফাংশন দেওয়া হলঃ

ফাংশন	বর্ণনা
`GETDATE()`	বর্তমান `date` এবং `time` রিটার্ন করবে।
`DATEPART()`	`date` অথবা `time` যেকোন একটি অংশ রিটার্ন করবে।
`DATEADD()`	`date` থেকে একটি নির্দিষ্ট সময় ব্যবধান যোগ/বিয়োগ করবে।
`DATEDIFF()`	দুইটি `date` এর পার্থক্য রিটার্ন করবে।
`CONVERT()`	`date` অথবা `time` কে দেখানোর জন্য বিভিন্ন ফরম্যাট নির্ধারন করবে।

SQL Date টাইপ

MySQL ডেটাবেজে `date` অথবা `date/time` এর সিনট্যাক্সঃ

- `DATE` - ফরম্যাটঃ `YYYY-MM-DD`
- `DATETIME` - ফরম্যাটঃ `YYYY-MM-DD HH:MI:SS`
- `DATETIME` - ফরম্যাটঃ `YYYY-MM-DD HH:MI:SS`
- `YEAR` - ফরম্যাটঃ `YYYY` অথবা `YY`

SQL Server ডেটাবেজে `date` অথবা `date/time` এর সিনট্যাক্সঃ

- `DATE` - ফরম্যাটঃ `YYYY-MM-DD`
- `DATETIME` - ফরম্যাটঃ `YYYY-MM-DD HH:MI:SS`
- `SMALLDATETIME` - ফরম্যাটঃ `YYYY-MM-DD HH:MI:SS`
- `DATETIME` - ফরম্যাটঃ একটি ইউনিক নাম্বার

বিঃদ্রঃ ডাটবেজে নতুন টেবিল তৈরি করার সময় আপনি কলাম এর জন্য `date` এর টাইপ নির্ধারণ করে দিতে পারবেন!

SQL Date ফাংশন এর ব্যবহার

তারিখের সাথে অন্য কোন উপাদান জড়িত না থাকলে আপনি খুব সহজেই দুইটি তারিখের মধ্যে তুলনা করতে পারেন।

নিম্নলিখিত "Student_attendance" টেবিলটি দেখুনঃ

আইডি নং	রোল নাম্বার	উপস্থিতি	ভর্তির তারিখ
১	১০১	৮৯	০১-১১-২০১৫
২	১০২	৯১	০১-১১-২০১৫
৩	১০৩	৮০	০১-১১-২০১৫
৪	১০৪	৭৫	০২-১১-২০১৫
৫	১০৫	৭৭	০২-১১-২০১৫

এখন আমরা "ভর্তির তারিখ(Admission_date)" কলাম থেকে "০১-১১-২০১৫" তারিখের রেকর্ড-সমূহ সিলেক্ট করবো।

উদাহরণ

```
SELECT * FROM Student_attendance
WHERE Admission_date='০১-১১-২০১৫';
Copy
```

ফলাফলঃ

আইডি নং	রোল নাম্বার	উপস্থিতি	ভর্তির তারিখ
১	১০১	৮৯	০১-১১-২০১৫
২	১০২	৯১	০১-১১-২০১৫
৩	১০৩	৮০	০১-১১-২০১৫

এখন আমরা "Student_attendance" টেবিলের "ভর্তির তারিখ(Admission_date)" কলামে তারিখের সাথে সময়ও যুক্ত করবো।

ধরুন, এখন আমাদের "Student_attendance" টেবিলটি নিম্নের মত দেখাবেঃ

আইডি নং	রোল নাম্বার	উপস্থিতি	ভর্তির তারিখ
১	১০১	৮৯	০১-১১-২০১৫ ১১:৩০:২০
২	১০২	৯১	০১-১১-২০১৫ ১১:৩২:১০
৩	১০৩	৮০	০১-১১-২০১৫ ১১:৪০:৪৫

এখন আমরা যদি উপরের মত একই `SELECT` স্টেটমেন্ট ব্যবহার করি তাহলে আমরা কোন ফলাফল পাবো না! কারন এই কুয়েরিটি শুধুমাত্র `date` কে সিলেক্ট করবে।

বিঃদ্রঃ আপনি যদি আপনার কুয়েরিকে সহজ রাখতে চান তাহলে `date` কলামে `time` কে ভিন্ন কলামে যোগ করুন।

SQL ডেটা টাইপ

একটি কলাম কোন ধরনের ভ্যালু ধারণ করবে তা ডেটা টাইপের মাধ্যমে ডিফাইন করা হয়।

SQL ডেটা টাইপ

ডেটাবেজ টেবিলের প্রতিটি কলামের জন্য একটি নাম এবং ডেটা টাইপ বাধ্যতামূলক।

টেবিল তৈরির সময়ে SQL ডেভেলপারদেরকে সিদ্ধান্ত নিতে হয় যে, প্রতিটি টেবিলের কলামে কোন টাইপের ডেটা সংরক্ষিত হবে। প্রতিটি কলামে কোন ধরনের ডেটা থাকবে তা ডেটা টাইপের মাধ্যমে SQL কে বুঝিয়ে দেওয়া হয় এবং এর মাধ্যমে SQL তার সংরক্ষিত ডেটা নিয়ে কিভাবে কাজ করবে তা বুঝতে পারে।

নিচের টেবিলে SQL এর সাধারণ ডেটা টাইপের একটি তালিকা দেওয়া হলঃ

ডেটা টাইপ	বর্ণনা
`CHARACTER(n)`	ক্যারেক্টার স্ট্রিং। **n** দ্বারা ক্যারেক্টার এর নির্দিষ্ট দৈর্ঘ্য নির্ধারণ করা হয়।
`VARCHAR(n)` **অথবা** `CHARACTER VARYING(n)`	ক্যারেক্টার স্ট্রিং। **n** দ্বারা ভ্যারিয়েবলের সর্বোচ্চ দৈর্ঘ্য নির্ধারণ করা হয়।
`BINARY(n)`	বাইনারী স্ট্রিং। **n** দ্বারা এর নির্দিষ্ট দৈর্ঘ্য নির্ধারণ করা হয়।
`BOOLEAN`	এর দ্বারা **TRUE** অথবা **FALSE** ভ্যালু সংরক্ষন করা হয়।
`VARBINARY(n)` **অথবা** `BINARY VARYING(n)`	বাইনারী স্ট্রিং। **n** দ্বারা ভ্যারিয়েবলের সর্বোচ্চ দৈর্ঘ্য নির্ধারণ করা হয়।
`INTEGER(p)`	নিউমেরিক ইন্টেজার(দশমিক নয়)। সর্বোচ্চ **p** সংখ্যা পর্যন্ত নির্ভুলভাবে দেখাবে।
`SMALLINT`	নিউমেরিক ইন্টেজার(দশমিক নয়)। সর্বোচ্চ **৫** সংখ্যা পর্যন্ত নির্ভুলভাবে দেখাবে।
`INTEGER`	নিউমেরিক ইন্টেজার(দশমিক নয়)। সর্বোচ্চ **১০** সংখ্যা পর্যন্ত নির্ভুলভাবে দেখাবে।
`BIGINT`	নিউমেরিক ইন্টেজার(দশমিক নয়)। সর্বোচ্চ **১৯** সংখ্যা পর্যন্ত নির্ভুলভাবে দেখাবে।
`DECIMAL(p,s)`	সুনির্দিষ্ট সংখ্যা, নির্ভুলভাবে **p** পর্যন্ত দেখাবে, স্কেল **s** । উদাহরনঃ `decimal(5,2)` একটি সংখ্যা যার দশমিকের পূর্বে **৩**টি ডিজিট থাকবে এবং দশমিকের পরে **২টি** ডিজিট থাকবে।
`NUMERIC(p,s)`	সুনির্দিষ্ট সংখ্যা, নির্ভুলভাবে **p** পর্যন্ত দেখাবে, স্কেল **s** । (`DECIMAL` এর মতই)
`FLOAT(p)`	সম্ভাব্য সংখ্যা, আংশিক নির্ভুলভাবে **p** পর্যন্ত দেখাবে। এই টাইপের জন্য সাইজ আর্গুমেন্টটি একটি সংখ্যা দ্বারা নির্ভুলতা সর্বনিম্ন কত সংখ্যা পর্যন্ত হবে তা নির্দেশ করে।
`REAL`	সম্ভাব্য সংখ্যা, **৭** পর্যন্ত আংশিক নির্ভুলভাবে প্রদর্শন করবে।/td>
`FLOAT`	সম্ভাব্য সংখ্যা, **১৬** পর্যন্ত আংশিক নির্ভুলভাবে প্রদর্শন করবে।
`DOUBLE PRECISION`	সম্ভাব্য সংখ্যা, **১৬** পর্যন্ত আংশিক নির্ভুলভাবে প্রদর্শন করবে।
`DATE`	বছর, মাস এবং দিনের ভ্যালু সংরক্ষন করে
`TIME`	ঘন্টা, মিনিট এবং সেকেন্ডের ভ্যালু সংরক্ষন করে
`TIMESTAMP`	বছর, মাস, দিন, ঘন্টা, মিনিট এবং সেকেন্ডের ভ্যালু সংরক্ষন করে
`INTERVAL`	একাধিক ইন্টেজার ফিল্ড এর সংমিশ্রণ যা ব্যবধির উপর ভিত্তিকরে একটি সময়কাল নির্ধারণ করে।
`ARRAY`	সেট এর দৈর্ঘ্য এবং ক্রমিকভাবে(ordered) তথ্য সংগ্রহকে বুঝায়`set-length` এবং `ordered` কালেকশন
`MULTISET`	ভ্যারিয়েবলের দৈর্ঘ্য এবং এলোমেলোভাবে(unorderd) তথ্য সংগ্রহকে বুঝায়
`XML`	এক্সএমএল ডেটা সংরক্ষন করে

SQL ফাংশন

SQL-এর অনেক নিজস্ব(Built-in) ফাংশন রয়েছে, যা তথ্য হিসাব-নিকাশে ব্যবহৃত হয়।

SQL Aggregate ফাংশন

`aggregate` ফাংশন একটি কলামের ভ্যালুগুলো হিসাব(calculation) করে একটি একক মান রিটার্ন করে।

প্রয়োজনীয় `aggregate` ফাংশনঃ

- `AVG()` - গড় মান রিটার্ন করে।
- `COUNT()` - সারি সংখ্যা রিটার্ন করে।
- `FIRST()` - প্রথম ভ্যালুটি রিটার্ন করে।
- `LAST()` - শেষ ভ্যালুটি রিটার্ন করে।
- `MAX()` - বৃহত্তম ভ্যালুটি রিটার্ন করে।
- `MIN()` - ক্ষুদ্রতম ভ্যালুটি রিটার্ন করে।
- `SUM()` - সমষ্টি রিটার্ন করে।

SQL Scalar ফাংশন

ইনপুট ভ্যালুর উপর ভিত্তি করে `scalar` ফাংশন একক মান রিটার্ন করে।

প্রয়োজনীয় `scalar` ফাংশনঃ

- `UCASE()` - একটি ফিল্ডের অক্ষর সমূহকে বড়-হাতের অক্ষরে রূপান্তর করে।
- `LCASE()` - একটি ফিল্ডের অক্ষর সমূহকে ছোট-হাতের অক্ষরে রূপান্তর করে।
- `MID()` - একটি টেক্সট ফিল্ডের থেকে অক্ষর সমূহকে নিষ্কাশন(extract) করে।
- `LEN()` - টেক্সট ফিল্ডের দৈর্ঘ্য কে রিটার্ন করে।
- `ROUND()` - দশমিক সংখ্যাকে একটি নির্দিষ্ট পূর্ণ সাংখ্যায় প্রকাশ করে।
- `NOW()` - সিস্টেমের বর্তমান তারিখ এবং সময় রিটার্ন করে।
- `FORMAT()` - একটি ফিল্ডকে প্রদর্শন করানোর গঠন(formats) নির্ধারন করে।

SQL AVG() ফাংশন

`AVG()` ফাংশন একটি কলামের গড় মান রিটার্ন করে।

SQL AVG() সিনট্যাক্স

নিম্নে SQL `AVG()` ফাংশনটির সিনট্যাক্স দেওয়া হলঃ

```
SELECT AVG(name_of_column)
FROM name_of_table;
Copy
```

নমুনা ডেটাবেজ

AVG() ফাংশনের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ Student ব্যবহার করবো।

SQL AVG() উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিলের "রোল নম্বর(Roll_number)" কলামের গড় মান নিয়ে আসবেঃ

উদাহরণ

```
SELECT AVG(Roll_number)
FROM Student_details;
Copy
```

SQL COUNT() ফাংশন

SQL COUNT() ফাংশনটি ডেটাবেজে নির্ধারিত বৈশিষ্ট্যের ভিত্তিতে সদৃশ(matched) সারির সংখ্যা রিটার্ন করে।

SQL COUNT(name_of_column) সিনট্যাক্স

COUNT(name_of_column) ফাংশনটি একটি নির্দিষ্ট কলামের মোট ভ্যালুর সংখ্যা রিটার্ন করে। তবে এক্ষেত্রে NULL ভ্যালুগুলো হিসাবযোগ্য(countable) নহেঃ

```
SELECT COUNT(name_of_column)
FROM name_of_table;
Copy
```

SQL COUNT(*) সিনট্যাক্স

COUNT(*) ফাংশনটি একটি টেবিলের সর্বমোট রেকর্ড সংখ্যা রিটার্ন করেঃ

```
SELECT COUNT(*)
FROM name_of_table;
Copy
```

SQL COUNT(DISTINCT name_of_column) সিনট্যাক্স

`COUNT(DISTINCT name_of_column)` ফাংশনটি একটি নির্দিষ্ট কলামের মোট স্বতন্ত্র(distinct) রেকর্ড সংখ্যা রিটার্ন করেঃ

```
SELECT COUNT(DISTINCT name_of_column)
FROM name_of_table;
Copy
```

বিঃদ্রঃ `COUNT(DISTINCT)` ফাংশনটি শুধুমাত্র Oracle এবং SQL Server এ কাজ করে কিন্তু MS Access এ কাজ করে না।

নমুনা ডেটাবেজ

`COUNT()` ফাংশনের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

SQL COUNT() উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিলের "রোল নাম্বার(Roll_number)"=১০৩ এ কতটি রেকর্ড রয়েছে তা গণনা করবেঃ

উদাহরণ

```
SELECT COUNT(Roll_number) AS StudentDetailsRoll_103
FROM Student_details
WHERE Roll_number=103;
Copy
```

SQL COUNT(*) উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিলের সকল রেকর্ডকে গণনা করবেঃ

উদাহরণ

```
SELECT COUNT(*) AS TotalStudentDetails
FROM Student_details;
Copy
```

SQL COUNT(DISTINCT name_of_column) উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিলের ইউনিক রেকর্ড গুলোকে গণনা করবেঃ

উদাহরণ

```
SELECT COUNT(DISTINCT Roll_number) AS TotalStudentDetails
FROM Student_details;
```

SQL FIRST() ফাংশন

SQL `FIRST()` ফাংশনটি নির্বাচিত(selected) কলামের প্রথম ভ্যালুটি রিটার্ন করে।

SQL FIRST() সিনট্যাক্স

```
SELECT FIRST(name_of_column)
FROM name_of_table;
Copy
```

বিঃদ্রঃ `FIRST()` ফাংশনটি শুধুমাত্র MS Access এ সাপোর্ট করে।

SQL FIRST() ফাংশন ব্যবহারের জন্য MySQL, Oracle এবং SQL Server এ ভিন্ন পদ্ধতি ব্যবহার করা হয়।

MySQL সিনট্যাক্স

```
SELECT name_of_column
FROM name_of_table
ORDER BY name_of_column ASC LIMIT 1;
Copy
```

Oracle সিনট্যাক্স

```
SELECT name_of_column
FROM name_of_table
WHERE ROWNUM <=1
ORDER BY name_of_column ASC;
Copy
```

SQL সার্ভার সিনট্যাক্স

```
SELECT TOP 1 name_of_column
FROM name_of_table
ORDER BY name_of_column ASC;
Copy
```

নমুনা ডেটাবেজ

`FIRST()` ফাংশনের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

SQL FIRST() ফাংশনের উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিলের "শিক্ষার্থীর নাম(Student_name)" কলামের প্রথম ভ্যালুটিকে সিলেক্ট করবেঃ

উদাহরণ

```
SELECT Student_name
FROM Student_details
ORDER BY Student_name ASC LIMIT 1;
```

SQL LAST() ফাংশন

SQL `LAST()` ফাংশনটি নির্বাচিত(selected) কলামের শেষ ভ্যালুটি রিটার্ন করে।

SQL LAST() সিনট্যাক্স

```
SELECT LAST(name_of_column)
FROM name_of_table;
Copy
```

বিঃদ্রঃ `LAST()` ফাংশনটি শুধুমাত্র MS Access এ সাপোর্ট করে।

SQL LAST() ফাংশন ব্যবহারের জন্য MySQL, Oracle এবং SQL Server এ ভিন্ন পদ্ধতি ব্যবহার করা হয়।

MySQL সিনট্যাক্স

```
SELECT name_of_column
FROM name_of_table
ORDER BY name_of_column DESC LIMIT 1;
Copy
```

Oracle সিনট্যাক্স

```
SELECT name_of_column
FROM name_of_table
ORDER BY name_of_column DESC
WHERE ROWNUM <=1;
Copy
```

SQL Server সিনট্যাক্স

```
SELECT TOP 1 name_of_column
FROM name_of_table
ORDER BY name_of_column DESC;
```

Collected By
Md. Jubayir Hossain

```
Copy
```

নমুনা ডেটাবেজ

`LAST()` ফাংশনের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

SQL LAST() ফাংশনের উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিলের "Student_name" কলামের শেষ ভ্যালুটিকে সিলেক্ট করবেঃ

উদাহরণ

```
SELECT Student_name
FROM Student_details
ORDER BY Student_name DESC LIMIT 1;
```

SQL MAX() ফাংশন

SQL `MAX()` ফাংশনটি নির্বাচিত(selected) কলামের বৃহত্তম ভ্যালুটি রিটার্ন করে।

SQL MAX() সিনট্যাক্স

```
SELECT MAX(name_of_column)
FROM name_of_table;
Copy
```

নমুনা ডেটাবেজ

`MAX()` ফাংশনের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_attendance" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	উপস্থিতি	ভর্তির তারিখ
১	১০১	৮৯	০১-১১-২০১৫
২	১০২	৯১	০১-১১-২০১৫
৩	১০৩	৮০	০১-১১-২০১৫
৪	১০৪	৭৫	০২-১১-২০১৫
৫	১০৫	৭৭	০২-১১-২০১৫

SQL MAX() ফাংশনের উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "Student_attendance" টেবিলের "উপস্থিতি(Attendance)" কলামের সবচেয়ে বৃহত্তম ভ্যালুটি নিয়ে আসবেঃ

উদাহরণ

```
SELECT MAX(Attendance) AS HighestAttendance
FROM Student_attendance;
Copy
```

SQL MIN() ফাংশন

`MIN()` ফাংশনটি নির্বাচিত(selected) কলামের ক্ষুদ্রতম ভ্যালুটি রিটার্ন করে।

SQL MIN() সিনট্যাক্স

```
SELECT MIN(name_of_column)
FROM name_of_table;
Copy
```

নমুনা ডেটাবেজ

`MIN()` ফাংশনের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_attendance" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	উপস্থিতি	ভর্তির তারিখ
১	১০১	৮৯	০১-১১-২০১৫
২	১০২	৯১	০১-১১-২০১৫
৩	১০৩	৮০	০১-১১-২০১৫
৪	১০৪	৭৫	০২-১১-২০১৫
৫	১০৫	৭৭	০২-১১-২০১৫

SQL MIN() ফাংশনের উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "Student_attendance" টেবিলের "উপস্থিতি(Attendance)" কলামের সবচেয়ে ক্ষুদ্রতম ভ্যালুটি নিয়ে আসবেঃ

উদাহরণ

```
SELECT MIN(Attendance) AS LowestAttendance
FROM Student_attendance;
Copy
```

SQL SUM() ফাংশন

SQL `SUM()` ফাংশনটি একটি কলামের সকল সংখ্যা ভ্যালুর সমষ্টি রিটার্ন করে।

SQL SUM() সিনট্যাক্স

```
SELECT SUM(name_of_column)
FROM name_of_table;
Copy
```

নমুনা ডেটাবেজ

`SUM()` ফাংশনের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_attendance" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	উপস্থিতি	ভর্তির তারিখ
১	১০১	৮৯	০১-১১-২০১৫
২	১০২	৯১	০১-১১-২০১৫
৩	১০৩	৮০	০১-১১-২০১৫

৪	১০৪	৭৫	০২-১১-২০১৫
৫	১০৫	৭৭	০২-১১-২০১৫

SQL SUM() ফাংশনের উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "Student_attendance" টেবিলের " উপস্থিতি(Attendance)" কলামের সকল ভ্যালুর সমষ্টি নিয়ে আসবেঃ

উদাহরণ

```
SELECT SUM(Attendance) AS TotalAttendance
FROM Student_attendance;
```

SQL UCASE() ফাংশন

SQL `UCASE()` ফাংশন একটি ফিল্ডের মানকে বড়-হাতের বর্ণে রূপান্তর করে।

SQL UCASE() সিনট্যাক্স

```
SELECT UCASE(name_of_column)
FROM name_of_table;
Copy
```

SQL Server এর জন্য সিনট্যাক্স

```
SELECT UPPER(name_of_column)
FROM name_of_table;
Copy
```

নমুনা ডেটাবেজ

`UCASE()` ফাংশনের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

SQL UCASE() ফাংশনের উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিল থেকে "শিক্ষার্থীর নাম(Student_name)" এবং "ঠিকানা(Address)" কলামকে সিলেক্ট করে "শিক্ষার্থীর নাম(Student_name)" কলামের ইংরেজী শব্দ গুলোকে বড়-হাতের বর্ণে রূপান্তর করবেঃ

উদাহরণ

```
SELECT UCASE(Student_name) AS Student, Address
FROM Student_details;
Copy
```

বিঃদ্রঃ `UCASE()` ফাংশন শুধুমাত্র ইংরেজী শব্দের সাথে কাজ করে।

SQL LCASE() ফাংশন

SQL `LCASE()` ফাংশনটি একটি ফিল্ডের মানকে ছোট-হাতের বর্ণে রূপান্তর করে।

SQL LCASE() সিনটেক্স

```
SELECT LCASE(name_of_column)
FROM name_of_table;
Copy
```

SQL Server এর জন্য সিনটেক্স

```
SELECT LOWER(name_of_column)
FROM name_of_table;
Copy
```

নমুনা ডেটাবেজ

LCASE() ফাংশনের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

SQL LCASE() ফাংশনের উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিল থেকে "শিক্ষার্থীর নাম(Student_name)" এবং "ঠিকানা(Address)" কলামকে সিলেক্ট করে "শিক্ষার্থীর নাম(Student_name)" কলামের ইংরেজী শব্দ গুলোকে ছোট-হাতের বর্ণে রূপান্তর করবেঃ

উদাহরণ

```
SELECT LCASE(Student_name) AS Student, Address
FROM Student_details;
Copy
```

বিঃদ্রঃ `LCASE()` ফাংশন শুধুমাত্র ইংরেজী শব্দের সাথে কাজ করে।

SQL MID() ফাংশন

SQL `MID()` ফাংশনটি টেক্সট ফিল্ড থেকে নির্দিষ্ট সংখ্যক বর্ণ নিষ্কাশন(extract) করে।

SQL MID() সিনটেক্স

```
SELECT MID(name_of_column, start, length) AS some_name
FROM name_of_table;
Copy
```

প্যারামিটার	বর্ণনা
`name_of_column`	আবশ্যক। যে ফিল্ড থেকে বর্ণ খুঁজে বের করে।
`start`	আবশ্যক। শুরুর অবস্থান নির্ধারণ করে।
`length`	ঐচ্ছিক। যত সংখ্যক অক্ষর রিটার্ন করবে তার দৈর্ঘ্য বুঝায়। যদি এটিকে বাদ দেওয়া হয় তাহলে `MID()` ফাংশনটি টেক্সট ফিল্ডের সকল অক্ষর গুলো নিয়ে আসে।

বিঃদ্রঃ SQL Server এ `MID()` ফাংশনের সমতুল্য ফাংশন হলো `SUBSTRING()` ফাংশন

```
SELECT SUBSTRING(name_of_column, start, length) AS some_name
FROM name_of_table;
Copy
```

নমুনা ডেটাবেজ

MID() ফাংশনের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

SQL MID() ফাংশনের উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিল থেকে "ঠিকানা(Address)" কলামের প্রথম তিনটি অক্ষর সিলেক্ট করবেঃ

উদাহরণ

```
SELECT MID(Address, 1, 3) AS ShortAddress
FROM Student_details;
Copy
```

ফলাফলঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	সংক্ষিপ্ত ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁ
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁ
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁ
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁ
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁ

SQL LEN() ফাংশন

`LEN()` ফাংশনটি টেক্সট ফিল্ডের মধ্যে অবস্থিত অক্ষরের সংখ্যা রিটার্ন করে।

SQL LEN() সিনট্যাক্স

```
SELECT LEN(name_of_column)
FROM name_of_table;
Copy
```

MySQL এর জন্য সিনট্যাক্স

MySQL-এ `LEN()` ফাংশনটি কাজ করে না এর পরিবর্তে আমরা `LENGTH()` ফাংশনটি ব্যবহার করবো।

```
SELECT LENGTH(name_of_column)
FROM name_of_table;
Copy
```

নমুনা ডেটাবেজ

LEN() ফাংশনের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

SQL LEN() ফাংশনের উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিল থেকে সকল কলাম সিলেক্ট করবে এবং "শিক্ষার্থীর নাম(Student_name)" কলামের অক্ষরের দৈর্ঘ্য নির্ণয় করে একটি নতুন কলামে ইনসার্ট করবেঃ

উদাহরণ

```
SELECT *, LENGTH(Student_name) AS LengthOfName
FROM Student_details;
Copy
```

ফলাফলঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	শিক্ষার্থীর নামের দৈর্ঘ্য	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	34	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	40	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	38	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	34	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	34	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

SQL ROUND() ফাংশন

ROUND() ফাংশনের উদাহরণ

SQL `ROUND()` ফাংশনটি নির্দিষ্ট দশমিক সংখ্যাকে একটি পূর্ণ সংখ্যায় প্রকাশ করতে ব্যবহৃত হয়।

SQL ROUND() সিনট্যাক্স

```
SELECT ROUND(name_of_column,decimals)
FROM name_of_table;
Copy
```

প্যারামিটার	বর্ণনা
name_of_column	আবশ্যক। যে ফিল্ডটি রাউন্ড করতে হবে।
decimals	আবশ্যক। দশমিক সংখ্যা ফেরত পাঠাবে।

নমুনা ডেটাবেজ

`ROUND()` ফাংশনের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_details" টেবিল থেকে নেওয়া হয়েছেঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১.০০	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২.০০	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩.০০	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪.০০	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

৫	১০৫.০০	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

SQL ROUND() ফাংশনের উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "Student_details" টেবিল থেকে "রোল নাম্বার(Roll_number)" কলামটি সিলেক্ট করে এর মান গুলোকে রাউন্ড করবেঃ

উদাহরণ

```
SELECT ROUND(Roll_number,0) AS Roll_number
FROM Student_details;
Copy
```

ফলাফলঃ

আইডি নং	রোল নাম্বার	শিক্ষার্থীর নাম	প্রতিষ্ঠানের নাম	ঠিকানা
১	১০১	তামজীদ হাসান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
২	১০২	মিনহাজুর রহমান	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৩	১০৩	মোঃ সবুজ হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৪	১০৪	ইয়াসিন হোসেন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর
৫	১০৫	ফরহাদ উদ্দিন	জাতীয় বিশ্ববিদ্যালয়	চাঁদপুর

QL NOW() ফাংশন

SQL `NOW()` ফাংশনটি সিস্টেমের বর্তমান সময় এবং তারিখ রিটার্ন করে।

SQL NOW() সিনটেক্স

```
SELECT NOW()
FROM name_of_table;
Copy
```

নমুনা ডেটাবেজ

NOW() ফাংশনের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_attendance" টেবিল থেকে নেওয়াঃ

আইডি নং	রোল নাম্বার	উপস্থিতি	ভর্তির তারিখ
১	১০১	৮৯	০১-১১-২০১৫
২	১০২	৯১	০১-১১-২০১৫
৩	১০৩	৮০	০১-১১-২০১৫
৪	১০৪	৭৫	০২-১১-২০১৫
৫	১০৫	৭৭	০২-১১-২০১৫

SQL NOW() ফাংশনের উদাহরণ

নিম্নের SQL স্টেটমেন্টটি "Student_attendance" টেবিল থেকে আজকের দিনে ভর্তিকৃত শিক্ষার্থীদের তথ্য নিয়ে আসবেঃ

উদাহরণ

```
SELECT NOW() AS TodaysAdmission
FROM Student_attendance;
Copy
```

SQL FORMAT() ফাংশন

SQL `FORMAT()` ফাংশনটি একটি ফিল্ডের তথ্য কিভাবে প্রদর্শিত হবে তা নির্ধারণ করে। ডেটাবেজ থেকে প্রাপ্ত তথ্যকে আমাদের ইচ্ছামত ফরম্যাটে দেখানোর জন্য আমরা `FORMAT()` ফাংশনটি ব্যবহার করবো।

SQL FORMAT() সিনট্যাক্স

```
SELECT FORMAT(name_of_column,format)
FROM name_of_table;
Copy
```

প্যারামিটার	বর্ণনা
`name_of_column`	আবশ্যক। যে ফিল্ডটি ফরম্যাট করতে হবে তা সিলেক্ট করবে।
`format`	আবশ্যক। ফরম্যাট নির্দেশ করে।

নমুনা ডেটাবেজ

`FORMAT()` ফাংশনের ব্যবহার দেখানোর জন্য আমরা আমাদের নমুনা ডেটাবেজ **Student** ব্যবহার করবো।

নিচের অংশটি "Student_attendance" টেবিল থেকে নেওয়াঃ

আইডি নং	রোল নাম্বার	উপস্থিতি	ভর্তির তারিখ
১	১০১	৮৯	০১-১১-২০১৫
২	১০২	৯১	০১-১১-২০১৫
৩	১০৩	৮০	০১-১১-২০১৫
৪	১০৪	৭৫	০২-১১-২০১৫
৫	১০৫	৭৭	০২-১১-২০১৫

SQL FORMAT() ফাংশনের ব্যবহার

নিম্নের SQL স্টেটমেন্টটি "Student_attendance" টেবিল থেকে আজকের দিনে ভর্তিকৃত শিক্ষার্থীর জন্য তারিখের একটি নির্দিষ্ট ফরম্যাট ঠিক করবেঃ

উদাহরণ

```
SELECT FORMAT(Now(),'YYYY-MM-DD') AS AdmittedDate
FROM Student_attendance;
```

SQL Temporary টেবিল

স্বল্প সময়ের জন্য ডেটাবেজে কোনো তথ্য জমা রাখতে SQL অস্থায়ী(Temporary) টেবিল ব্যবহার করা হয়।

SQL Temporary টেবিলের ব্যবহার

- অস্থায়ী টেবিলের অনেক ধরণের সুবিধা রয়েছে। RDBMS ডেটাবেজ সিস্টেমে অস্থায়ী টেবিল সাপোর্ট করে।
- অস্থায়ী টেবিল ব্যবহার করে আপনি ডেটাবেজ টেবিলে তথ্য জমা রাখার পাশাপাশি অন্যান্য টেবিলের মত `SELECT`, `UPDATE`, `JOIN` ইত্যাদি SQL কমান্ড সম্পাদন করতে পারবেন।
- অস্থায়ী তথ্য সংরক্ষনের জন্য কিছু কিছু ক্ষেত্রে অস্থায়ী টেবিল অতীব প্রয়োজন হয়ে পড়ে। যেমন- বর্তমান ইউজারের শেষন(session) সমাপ্ত হবে এমন সময় তার সকল তথ্য মুছে ফেলার জন্য অস্থায়ী টেবিল ব্যবহৃত হয়। MySql ভার্সন ৩.২৩ এবং পরবর্তী ভার্সন-সমূহে অস্থায়ী টেবিল যুক্ত হয়েছে।

বিঃদ্রঃ যতক্ষন পর্যন্ত শেষন(session) সক্রিয় থাকে কেবল ততক্ষন পর্যন্তই অস্থায়ী টেবিলে তথ্য সংরক্ষিত থাকে।

আপনি যদি পিএইচপি স্ক্রিপ্ট এর মাধ্যমে SQL কোড রান করান তাহলে পিএইচপি কোড সম্পাদন শেষে অস্থায়ী(Temporary) টেবিলে সংরক্ষিত তথ্য স্বয়ংক্রিয়ভাবে মুছে যাবে।

আপনি যদি MySQL ডেটাবেজকে এক্সেস(Access) করার জন্য MySql Client Program ব্যবহার করে থাকেন তাহলে প্রোগ্রামটি বন্ধ হওয়ার পরে অস্থায়ী টেবিলটি মুছে যাবে। অথবা আপনি নিজ থেকে ম্যানুয়ালিও টেবিলটি মুছে ফেলতে পারেন।

SQL Temporary টেবিলের সিনটেক্স

```
CREATE TEMPORARY TABLE name_of_table (
    name_of_column,
    name_of_column,
...
);
Copy
```

নিম্নের উদাহরনে আমরা অস্থায়ী টেবিলের ব্যবহার দেখবোঃ

```
CREATE TEMPORARY TABLE Student_works (
    Student_name VARCHAR(50) NOT NULL,
    Total_submited_work VARCHAR(50) NOT NULL
);
Copy
```

অস্থায়ী টেবিল গুলো আমরা স্বল্প প্রয়োজনে ব্যবহার করবো। এতে ডেটাবেজে জায়গাও সাশ্রয় হয়।

আপনি ডেটাবেজে কখনই অস্থায়ী টেবিলের তথ্য দেখতে পারবেন না। আপনি যদি **SHOW TABLES** কমান্ড ব্যবহার করে আপনার টেবিল লিষ্ট বের করেন সেখানেও অস্থায়ী টেবিলটি খুঁজে পাবেন না। এখন আপনি যদি লগ-আউট করেন অর্থাৎ আপনার বর্তমান MySql সেশনটি শেষ হওয়ার পর আপনি আর এই টেবিলের তথ্য গুলো অ্যাক্সেস(Access) করতে পাবেন না। এমন কি আপনার অস্থায়ী টেবিলটিও আর থাকবে না।

অস্থায়ী(temporary) টেবিল ডিলিট

MySql ডেটাবেজের ক্ষেত্রে ডেটাবেজ কানেকশনের সমাপ্তি ঘটলেই অস্থায়ী টেবিল মুছে যায়। কিন্তু আপনি যাদি ডেটাবেজের কাজ সম্পন্ন হওয়ার পূর্বেই অস্থায়ী টেবিল মুছে ফেলতে চান তাহলে **DROP TABLE** কমান্ড ব্যবহার করতে পারেন।

```
DROP TABLE name_of_table;
```

SQL Injection

SQL Injection হলো কোড ইঞ্জেক্ট করার একটি কৌশল যা আপনার ডেটাবেজকে ধ্বংস করে দিতে পারে।

ওয়েবপেজ এর মাধ্যমে ডেটাবেজে খারাপ(malicious) কোড ইনপুট/জমা করে রাখাই হলো SQL Injection।

ওয়েব সাইট হ্যাকিং কৌশলগুলোর মধ্য **SQL Injection** অন্যতম।

ওয়েব পেজে SQL এর ব্যবহার

পূর্ববর্তী অধ্যায়গুলোতে আমরা শিখে এসেছি SQL ব্যবহার করে কিভাবে ডেটাবেজের তথ্য আপডেট এবং পুনরূদ্ধার করতে হয়।

ওয়েব পেজে ইউজারদেরকে প্রায়ই তাদের নিজস্ব সার্চ ভ্যালু ব্যবহার করার অনুমতি দেওয়া হয় যা ব্যবহার করে তারা ওয়েব পেজের তথ্য প্রদর্শন করতে পারে।

SQL স্টেটমেন্ট-সমূহ শুধুমাত্র টেক্সট হওয়ায় ইউজার খুব সহজেই ডায়নামিকভাবে SQL কোড গুলোকে পরিবর্তন করে ফেলতে পারেঃ

ওয়েবপেজ থেকে আপনি যখন ইউজারকে তথ্য ইনপুট এর সুযোগ দেন কেবল তখনই SQL Injection ঘটে। যেমন- আপনি ইউজারকে তার নাম/আইডি ইনপুট করতে বললেন, কিন্তু সে নাম/আইডি এর পরিবর্তে ইনপুট হিসাবে নিম্নের ন্যায় SQL কমান্ড ইনপুট দিল এবং যা আপনি আপনার নিজের অজান্তেই আপনার ডেটাবেজে রান করালেন।

উদাহরণ

```
solidUserId = getRequestString("User_Id");
solidSQL = "SELECT * FROM Total_Users
WHERE User_Id = " + solidUserId;
Copy
```

উপরের উদাহরণে সিলেক্ট স্ট্রিং এর সাথে **solidUserId** ভ্যারিয়েবলটি যোগ করে একটি `SELECT` স্টেটমেন্ট তৈরি করা হয়েছে যা ইউজার এর নিকট হতে তথ্য ইনপুট নিয়ে ডেটাবেজ থেকে সংশ্লিষ্ট তথ্য নিয়ে আসে।

আবারও SQL Injection

SQL `Injection` এমন একটি কৌশল যা ওয়েব পেজের মাধ্যমে ডেটাবেজে খারাপ কোড ইনপুট দিয়ে SQL কমান্ড ইঞ্জেক্ট করতে পারে।

ইঞ্জেক্টকৃত SQL কমান্ড পূর্ববর্তী SQL কমান্ড গুলোকে পরিবর্তন করতে পারে যা ওয়েব এপ্লিকেশনের নিরাপত্তায় বিঘ্ন সৃষ্টি করে।

SQL Injection 1=1 এর উপর ভিত্তি করে সর্বদাই সত্য

উপরের উদাহরণটিতে আরো একবার লক্ষ্য করুন, ঐ কোডের মূল উদ্দেশ্য ছিল ইউজার আইডির মাধ্যমে একটি ইউজারকে সিলেক্ট করার জন্য SQL স্টেটমেন্ট তৈরি করা।

যদি ইউজারকে **ভুল** ইনপুট প্রদানে বাধা দেওয়া না হয়, তাহলে ইউজার নিম্নের মত কিছু **স্মার্ট** ইনপুট প্রবেশ করাবেঃ

User_Id:
103 or 1=1

সার্ভারের ফলাফল

```
SELECT * FROM Total_Users WHERE User_Id = 103 or 1=1
Copy
```

উপরের কোড একটি বৈধ SQL স্টেটমেন্ট। যেহেতু **WHERE 1 = 1** সর্বদাই সত্য, সুতরাং ইহা ইউজার টেবিল থেকে সকল তথ্য রিটার্ন করবে।

উপরের উদাহরণটি কি আপনার কাছে বিপজ্জনক বলে মনে হচ্ছে?
একবার ভেবে দেখুন যদি এই টেবিলে ইউজার এর নাম এবং পাসওয়ার্ড থাকতো তাহলে কি হত?

নিম্নের SQL স্টেটমেন্টি অনেকটা উপরের স্টেটমেন্ট এর মতইঃ

```
SELECT User_Id, User_Name, User_Pass
FROM Total_Users WHERE User_Id = 103 or 1=1
Copy
```

এক্ষেত্রে একজন স্মার্ট হ্যাকার ইনপুট ফিল্ডে 103 অথবা 1=1 ইনপুট করে ডেটাবেজ থেকে খুব সহজেই ইউজারের সকল তথ্য অ্যাক্সেস করতে পারবে।

SQL Injection ""="" এর উপর ভিত্তিকরে সর্বদাই সত্য

ইউজার লগ-ইন ভেরিফাই করার জন্য নিম্নে একটি সাধারন এইচটিএমএল ফর্ম এর গঠন দেওয়া হলোঃ

User Name:

Password:

Collected By
Md. Jubayir Hossain

সার্ভার কোড

```
User_Name = getRequestString("User_Name");
User_Pass = getRequestString("User_Pass");
sql = "SELECT * FROM Total_Users WHERE User_Name ='" + User_Name + "'
     AND User_Pass ='" + User_Pass + "'"
Copy
```

এক্ষেত্রে একজন স্মার্ট হ্যাকার ইনপুট ফিল্ডে " " **OR** " "=" " ইনপুট করে খুব সহজেই ডেটাবেজ থেকে ইউজারের সকল তথ্য অ্যাক্সেস করতে পারে।

সার্ভার কোড নিম্নের মত একটি বৈধ SQL স্টেটমেন্ট তৈরি করবেঃ

ফলাফল

```
SELECT * FROM Total_Users WHERE User_Name ="" or ""=""
AND User_Pass ="" or ""=""
Copy
```

উপরের স্টেটমেন্টটি একটি বৈধ SQL স্টেটমেন্ট। এটি টেবিল থেকে ইউজারের সকল তথ্য রিটার্ন করবে যেখানে **WHERE** ""="" সর্বদাই সত্য।

Batched SQL স্টেটমেন্ট এর উপর ভিত্তিকরে SQL Injection

অধিকাংশ ডেটাবেজই ব্যাচ SQL স্টেটমেন্ট সাপোর্ট করে। দুই বা ততোধিক SQL স্টেটমেন্ট এর সমষ্টিই হলো ব্যাচ SQL স্টেটমেন্ট। একটি স্টেটমেন্ট থেকে অন্য একটি স্টেটমেন্টকে আলাদা করতে সেমিকোলন ব্যবহার করা হয়।

উদাহরণ

```
SELECT * FROM Total_Users;
DROP TABLE Users_Student;
Copy
```

উপরের SQL স্টেটমেন্টটি "Total_Users" টেবিলের সকল তথ্য রিটার্ন করবে এবং "Users_Student" টেবিলটিকে ডিলেট করে দিবে।

ধরুন, নিম্নের ন্যায় আমাদের সার্ভার কোড আছেঃ

সার্ভার কোড

```
solidUserId = getRequestString("User_Id");
solidSQL = "SELECT * FROM Total_Users WHERE User_Id = " + solidUserId;
Copy
```

এবং একজন ইউজার নিম্নের মত ইনপুট দিলঃ

User id:
```
103; DROP
```

তাহলে সার্ভারে নিম্নের মত একটি বৈধ SQL স্টেটমেন্ট তৈরি হবেঃ

ফলাফল

```
SELECT * FROM Total_Users WHERE UserId = 103;
DROP TABLE Users_Student;
Copy
```

সুরক্ষার জন্য প্যারামিটার ব্যবহার করুন

কোনো কোনো ওয়েব ডেভেলোপার কিছু নির্দিষ্ট শব্দ বা ক্যারেক্টারকে "blacklist" করে রাখে। হ্যাকাররা যেন এই শব্দ বা ক্যারেক্টার গুলো ইনপুট ফিল্ডের সার্স প্যারামিটার হসাবে ব্যবহার করে SQL ইনজেকশন ঘটাতে না পারে।

ইহা খুব একটা কার্যকরী পদ্ধতি নয়। কিছু শব্দ যেমন- `delete` অথবা `drop` এবং কিছু ক্যারেক্টার যেমন- সেমিকোলন(;) অথবা উদ্‌ধৃতি চিহ্ন("") সকল SQL ভাষাতেই ব্যবহত হয়। তাই এই শব্দ বা ক্যারেক্টারসমূহ ইনপুটে সম্মতি দেওয়া উচিৎ।

SQL ইনজেকশন আক্রমণকে প্রতিরোধ করার সবচেয়ে কার্যকরী এবং সঠিক পদ্ধতি হলো SQL প্যারামিটার ব্যবহার করা।

SQL প্যারামিটার হলো ভ্যালু যা SQL কুয়েরি সম্পাদনের সময় যুক্ত করা হয়। ইহা SQL এর কন্ট্রোলার হিসাবে কাজ করে।

SQL হোস্টিং

SQL হোস্টিং

আপনি যদি আপনার ওয়েবসাইটের মাধ্যমে ডেটাবেজে ডেটা সংরক্ষন করে পুনরায় পেতে চান তাহলে আপনার ওয়েব সার্ভার SQL ভাষা ব্যবহার করে এমন একটি ডেটাবেজে-সিস্টেমে এক্সেস থাকতে হবে।

আপনি যদি Internet Service Provider(ISP) কর্তৃক আপনার ওয়েব সার্ভার হোস্ট করে থাকেন তাহলে আপনাকে SQL হোস্টিং প্ল্যান খুঁজতে হবে। MySQL, SQL Server, MS SQL Server এবং MS Access ইত্যাদি রিলেশনাল ডেটাবেজ-সমূহ সচারচর ব্যবহত SQL হোস্টিং ডেটাবেজ।

MySQL

MySQL ওয়েবসাইটের জন্য জনপ্রিয় এবং অধিক ট্রাফিকযুক্ত একটি ডেটাবেজ সফটওয়্যার।

MySQL অনেক শক্তিশালী এবং সম্পূর্ন বৈশিষ্ট্যযুক্ত SQL ডেটাবেজ সিস্টেম।

ওরাকল

ডেটাবেজ চালিত অধিক ট্রাফিকযুক্ত ওয়েবসাইটের জন্য ওরাকলও একটি জনপ্রিয় ডেটাবেজ সফটওয়্যার।

ওরাকল অনেক শক্তিশালী এবং সম্পূর্ন বৈশিষ্ট্যযুক্ত SQL ডেটাবেজ সিস্টেম।

MS SQL সার্ভার

ডেটাবেজ চালিত অধিক ট্রাফিকযুক্ত ওয়েবসাইটের জন্য মাইক্রোসফট SQL সার্ভার একটি জনপ্রিয় ডেটাবেজ সফটওয়্যার।

SQL সার্ভার অনেক শক্তিশালী এবং সম্পূর্ন বৈশিষ্ট্যযুক্ত SQL ডেটাবেজ সিস্টেম।

এক্সেস

মাইক্রোসফট এক্সেস সাধারণ ডেটাবেজের জন্য একটি ভালো সমাধান।

অধিক ট্রাফিকযুক্ত ওয়েব সাইটের জন্য এক্সেস ভালো কাজ করে না এবং ইহা MySQL, SQL সার্ভার অথবা ওরাকলের মত শক্তিশালী না।