

CONTENT

Introduction to DMBSs	1
Structured Query Language – SQL	4
MySQL and MariaDB	6
SQL interfaces	7
Installing MySQL	8
Installing and setting up WAMP	12
Creating databases with CLI	13
Creating databases with GUI	14
SQL basics	16
SQL data types	17
Classification of SQL codes	19
ALTER TABLE command	20
INSERT INTO statement	23
UPDATE statement	25
DELETE statement	26
SELECT statement	27
Comparison operators	28
SQL Wildcards	31
Logical operators	33
ORDER BY keyword	34
GROUP BY statement	35
HAVING clause	36
DISTINCT statement	37
SQL functions	37
Aliases	39
SQL Joins	40
Integrity constraints	42
SQL constraints	
NOT NULL	43
UNIQUE	45
DEFAULT	47
CHECK	49
PRIMARY KEY	51
FOREIGN KEY	54

Introduction to DBMS

Database Management Systems (DBMS) are software tools that provide an interface for interacting with databases.

දත්ත සමුදා කළමනාකරණ පද්ධති (DBMS) යනු දත්ත සමුදායන් සමඟ අන්තර්ක්‍රියා කිරීම සඳහා අතුරු මුහුණතක් සපයන මෘදුකාංග මෙවලම් වේ.

Allow users to perform operations such as inserting, updating, deleting, and querying data efficiently.

පරිශීලකයින්ට දත්ත ඇතුළත් කිරීම, යාවත්කාලීන කිරීම, මකා දැමීම සහ විමසීම් වැනි මෙහෙයුම් කාර්යක්ෂමව සිදු කිරීමට ඉඩ සලසයි.

Among the most influential DBMS are MySQL and its open-source successor, MariaDB.

වඩාත්ම බලගතු DBMS අතර MySQL සහ එහි විවෘත-මූලාශ්‍ර අනුප්‍රාප්තිකයා වන MariaDB වේ,

History of Database Management Systems

දත්ත සමුදා කළමනාකරණ පද්ධති පිළිබඳ ඉතිහාසය

Pre-1960s: Flat Files

1960 ට පෙර: පැතලි ගොනු

1960 – 1970: Hierarchical and Network Models

ධූරාවලි සහ ජාල ආකෘති

1970 – 1980: Relational Database Model

1980 – 1990: Commercial RDBMS

1990 – 2000: Open-Source DBMS

විවෘත මූලාශ්‍ර DBMS

2000 – 2010: NoSQL and Big Data

2010s and Beyond: Cloud and Hybrid Databases

වලාකුළු සහ දෙමුහුන් දත්ත සමුදායන්

Further discussed in the database tute.
දත්ත සමුදා නිබන්දනයේ වැඩිදුර සාකච්ඡා කර ඇත.

Types of Database Management Systems

දත්ත සමුදා කළමනාකරණ පද්ධති වර්ග

1. Relational DBMS (RDBMS)

Organizes data into structured tables (e.g: MySQL, MariaDB, PostgreSQL).

දත්ත ව්‍යුහගත වගු වලට සංවිධානය කරයි (උදා: MySQL, MariaDB, PostgreSQL).

2. NoSQL DBMS

Designed for unstructured or semi-structured data (e.g: MongoDB, Cassandra). ව්‍යුහගත නොවන හෝ අර්ධ ව්‍යුහගත දත්ත සඳහා නිර්මාණය කර ඇත (උදා: MongoDB, Cassandra).

3. In-Memory DBMS

Optimized for high-speed, real-time processing (e.g: Redis, Memcached). අධිවේගී, තත්ත්ව කාලීන සැකසුම් සඳහා ප්‍රශස්ත කර ඇත (උදා: Redis, Memcached)

4. Cloud-Based DBMS

Managed, scalable databases hosted in the cloud (e.g., Firebase, Amazon RDS, Google Cloud Firestore).

වලාකුළු පරිගණක තුල සංරක්ෂණය වන, කළමනාකරණය කළ හැකි සහ විශාල කළ හැකි දත්ත ගබඩා. (උදා: Firebase, Amazon RDS, Google Cloud Firestore).

5. Hybrid DBMS

Combines SQL and NoSQL features for flexibility (e.g: Couchbase, Cosmos DB).

නම්‍යශීලීතාවය සඳහා SQL සහ NoSQL විශේෂාංග ඒකාබද්ධ කරයි (උදා: Couchbase, Cosmos DB).

6. Object-Oriented DBMS

Supports object-oriented programming paradigms.

වස්තු නැඹුරු වැඩසටහන් සුසාම්‍යභාවය සඳහා සහය දක්වයි.

Use cases of DBMSs

DBMS භාවිතා කරන අවස්ථා

1. Data Analysis and Reporting

දත්ත විශ්ලේෂණය සහ වාර්තා කිරීම

- Storing and managing large volumes of data for analysis.
විශ්ලේෂණය සඳහා විශාල දත්ත පරිමාවක් ගබඩා කිරීම සහ කළමනාකරණය කිරීම.
- Creating reports to support decision making.
භීරණ ගැනීමට සහාය වීම සඳහා වාර්තා නිර්මාණය කිරීම.
- Discovering patterns and trends in data.
දත්තවල රටා සහ ප්‍රවණතා සොයා ගැනීම.

2. Human Resources

මානව සම්පත්

- Storing employee information, salaries, benefits, and performance data.
සේවක තොරතුරු, වැටුප්, ප්‍රතිලාභ සහ කාර්ය සාධන දත්ත ගබඩා කිරීම.
- Managing job applications, candidate tracking, and onboarding.
රැකියා අයදුම්පත් කළමනාකරණය, අපේක්ෂකයින් ලුහුබැඳීම සහ ඇතුල්වීම.
- Processing employee salaries, taxes, and deductions.
සේවක වැටුප්, බදු සහ අඩු කිරීම් සැකසීම.

3. Business and Finance

ව්‍යාපාර සහ මූල්‍ය

- Generating financial statements, balance sheets, and income statements.
මූල්‍ය ප්‍රකාශන, ශේෂ පත්‍ර සහ ආදායම් ප්‍රකාශන ජනනය කිරීම.
- Managing customer data, tracking sales, and analyzing customer behavior.
පාරිභෝගික දත්ත කළමනාකරණය කිරීම, විකුණුම් නිරීක්ෂණය කිරීම සහ පාරිභෝගික හැසිරීම විශ්ලේෂණය කිරීම.

4. Healthcare

සෞඛ්‍ය සත්කාර

- Storing and retrieving patient information, medical history, and treatment details.
රෝගියාගේ තොරතුරු, වෛද්‍ය ඉතිහාසය සහ ප්‍රතිකාර විස්තර ගබඩා කිරීම සහ ලබා ගැනීම.
- Managing patient data, tracking outcomes, and analyzing research results.
රෝගීන්ගේ දත්ත කළමනාකරණය කිරීම, ප්‍රතිඵල නිරීක්ෂණය කිරීම සහ පර්යේෂණ ප්‍රතිඵල විශ්ලේෂණය කිරීම.
- Identifying patient trends, optimizing resources, and enhancing care.
රෝගීන්ගේ ප්‍රවණතා හඳුනා ගැනීම, සම්පත් ප්‍රශස්ත කිරීම සහ රැකවරණය වැඩි දියුණු කිරීම.

5. Education

අධ්‍යාපන

- Managing student records, course information, and enrollment data.
ශිෂ්‍ය වාර්තා කළමනාකරණය, පාඨමාලාව තොරතුරු, සහ ලියාපදිංචි දත්ත.

6. Government

ආණ්ඩුව

- Storing citizen information, tax data, and government records.
පුරවැසි තොරතුරු, බදු දත්ත සහ රජයේ වාර්තා ගබඩා කිරීම.

7. Telecommunications

විදුලි සංදේශ

- Managing customer data, network information, and billing.
පාරිභෝගික දත්ත, ජාල තොරතුරු සහ බිල්පත් කළමනාකරණය.

Modern Database Management Systems හවිත දත්ත සමුදා කළමනාකරණ පද්ධති

1. Firebase (Google)

A NoSQL, serverless cloud database for real-time mobile and web app data synchronization.

තත්‍ය කාලීන ජංගම සහ වෙබ් යෙදුම් දත්ත සමමුහුර්ත කිරීම සඳහා වන NoSQL සේවාදායක රහිත වලාකුළු දත්ත ගබඩාවකි.

2. Google BigQuery

A serverless data warehouse optimized for large-scale analytics and querying massive datasets.

මහා පරිමාණ විශ්ලේෂණ සහ දැවැන්ත දත්ත කට්ටල විමසීම සඳහා ප්‍රශස්ත කළ සේවාදායක රහිත දත්ත ගබඩාවකි.

3. Amazon DynamoDB

A NoSQL database offering extreme scalability and reliability, often used in IoT and real-time analytics.

IoT සහ තත්‍ය කාලීන විශ්ලේෂණවල බොහෝ විට භාවිතා කරන අතිශය පරිමාණය සහ විශ්වසනීයත්වය සපයන NoSQL දත්ත ගබඩාවකි.

4. Couchbase

A hybrid NoSQL database combining document, key-value, and relational features for scalable, distributed systems.

පරිමාණය කළ හැකි, බෙදා හරින ලද පද්ධති සඳහා ලේඛනය, යතුරු අගය සහ සම්බන්ධතා විශේෂාංග ඒකාබද්ධ කරන දෙමුහුන් NoSQL දත්ත සමුදායකි.

5. Azure Cosmos DB

A globally distributed, multi-model database supporting APIs like SQL, MongoDB, and Cassandra.

SQL, MongoDB සහ Cassandra වැනි API සහය දක්වන, ගෝලීය වශයෙන් විහිදී ඇති, බහු-ආදර්ශ දත්ත ගබඩාවකි.

6. MongoDB

A document-oriented NoSQL database storing JSON-like data for flexibility.

හමුසැලි බව සඳහා JSON වැනි දත්ත ගබඩා කරන ලේඛන-නැඹුරු NoSQL දත්ත ගබඩාවකි.

7. Neo4j

A graph database for managing interconnected data, useful in fraud detection and recommendation engines.

අන්තර් සම්බන්ධිත දත්ත කළමනාකරණය සඳහා වන ප්‍රස්ථාර දත්ත සමුදායකි, වංචා හඳුනාගැනීම් සහ නිර්දේශ වින්පින් සඳහා ප්‍රයෝජනවත් වේ.

Evolution of Query Languages

විමසුම් භාෂා පරිණාමය

SQL- The 1970s relational standard for structured data with DDL, DML, and DCL commands.

1970 දශකයේ DDL, DML සහ DCL විධාන සමඟ ආකෘතිමය දත්ත සඳහා වන සම්බන්ධක සම්මතයයි.

NoSQL Query Languages- JSON-based APIs for schema-less data (e.g: MongoDB, Firebase). schema -සටහන නොමැති දත්ත සඳහා JSON -මූලික APIs (උදා: MongoDB, Firebase)

Graph Query Languages- Specialized languages like Cypher for querying graph data (e.g: Neo4j).

ප්‍රස්ථාර දත්ත සඳහා Cypher වැනි විශේෂිත භාෂා (උදා: Neo4j)

SQL - Structured Query Language

SQL is a standard programming language used for managing and manipulating relational databases.

SQL යනු සම්බන්ධතා දත්ත සමුදායන් කළමනාකරණය කිරීම සහ හැසිරවීම සඳහා භාවිතා කරන සම්මත ක්‍රමලේඛන භාෂාවකි.

Birth of SQL

SQL's journey began in the 1970s, driven by the need for a more efficient way to interact with relational databases.

SQL හි ගම්‍ය ආරම්භ වූයේ 1970 දේදී, සම්බන්ධතා දත්ත සමුදායන් සමඟ අන්තර් ක්‍රියා කිරීමට වඩාත් කාර්යක්ෂම ක්‍රමයක් අවශ්‍ය වීමෙනි.

At IBM, researchers Donald D. Chamberlin and Raymond F. Boyce, inspired by Edgar F. Codd's relational data model, developed SEQUEL in the early 1970s.

IBM හි, පර්යේෂකයන් දෙදෙනෙකු වන ඩොනල්ඩ් ඩී. චේම්බර්ලින් සහ රේමන්ඩ් ඩී. බෝයිස්, එඩ්ගාර් ඩී. කොඩ්ගේ සම්බන්ධතා දත්ත ආකෘතියෙන් ආභාෂය ලැබූ අතර, 1970 ගණන්වල මුල් භාගයේදී SEQUEL සංවර්ධනය කරන ලදී.

The name SEQUEL was later shortened to the more concise SQL.

SEQUEL යන නම පසුව වඩාත් සංක්ෂිප්ත SQL ලෙස කෙටි කරන ලදී.

Oracle company became the first company to offer a commercial RDBMS (relational Database management system) that utilized SQL.

Oracle සමාගම SQL භාවිතා කරන වාණිජ RDBMS(සම්බන්ධතා දත්ත සමුදා කළමනාකරණ පද්ධති) ලබා දුන් පළමු සමාගම බවට පත් විය.

Different versions of SQL

SQL හි විවිධ සංස්කරණ

Microsoft SQL Server (Transact-SQL)

Offers proprietary extensions and features beyond standard SQL.

සම්මත SQL ඉක්මවා හිමිකම් සහිත දිගු සහ විශේෂාංග පිරිහමයි.

MySQL

A widely used open-source DBMS with its own SQL syntax.

ස්වකීය SQL වාක්‍ය ඛණ්ඩයක් සහිත පුළුල් ලෙස භාවිතා වන විවෘත මූලාශ්‍ර DBMS එකකි.

Oracle Database (PL/SQL)

Includes procedural extensions and advanced data types.

ක්‍රියා පටිපාටි දිගු කිරීම් සහ උසස් දත්ත වර්ග ඇතුළත් වේ.

PostgreSQL

An open-source object-relational DBMS with strong SQL compliance.

ශක්තිමත් SQL අනුකූලතාවයක් සහිත විවෘත මූලාශ්‍ර වස්තු-සම්බන්ධතා DBMS.

SQLite

A lightweight, embedded database with a simplified SQL implementation.

සරල කළ SQL ක්‍රියාත්මක කිරීමක් සහිත සැහැල්ලු, කාවැද්දූ දත්ත සමුදායක් සංස්කරණයක් වේ.

Why Use SQL?

SQL භාවිතා කරන්නේ ඇයි?

Data Management

දත්ත කළමනාකරණය

SQL is the standard language for creating, modifying, and deleting database structures.

SQL යනු දත්ත සමුදා ව්‍යුහයන් නිර්මාණය කිරීම, වෙනස් කිරීම සහ මකා දැමීම සඳහා වන සම්මත භාෂාවයි.

Data Retrieval

දත්ත ලබා ගැනීම

It allows you to extract specific information from the database using powerful query capabilities.

බලගතු විමසුම් හැකියාවන් භාවිතයෙන් දත්ත සමුදායෙන් නිශ්චිත තොරතුරු උකහා ගැනීමට එය ඔබට ඉඩ සලසයි.

Data Updates

දත්ත යාවත්කාලීන කිරීමට

You can modify existing data records efficiently. ඔබට පවතින දත්ත වාර්තා කාර්යක්ෂමව වෙනස් කළ හැක.

Data Integrity

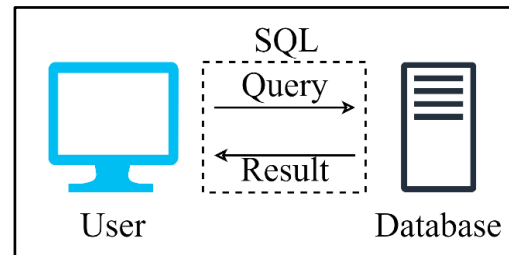
දත්ත අඛණ්ඩතාව

SQL helps maintain data consistency and accuracy.

SQL දත්ත අනුකූලතාව සහ නිරවද්‍යතාවය පවත්වා ගැනීමට උපකාරී වේ.

How SQL Works

SQL ක්‍රියා කරන ආකාරය



An SQL query is written

SQL විමසුමක් ලියනු ලැබේ

A command or question is posed to the database.

දත්ත සමුදාය වෙත විධානයක් හෝ ප්‍රශ්නයක් ඉදිරිපත් කෙරේ.

The query is parsed by the SQL engine

විමසුම SQL එන්ජින් මගින් විග්‍රහ කරනු ලැබේ

The query is broken down into understandable components.

විමසුම තේරුම් ගත හැකි සංරචක වලට බෙදා ඇත.

The query is optimized

විමසුම ප්‍රශස්ත ලෙස සකසනු ලැබේ

The most efficient way to execute the query is determined by the database system.

විමසුම ක්‍රියාත්මක කිරීමේ වඩාත් කාර්යක්ෂම ක්‍රමය දත්ත සමුදා පද්ධතිය මගින් තීරණය වේ.

The necessary data is retrieved

අවශ්‍ය දත්ත නැවත ලබා ගනී

The system accesses the required data from the tables.

වගු වලින් අවශ්‍ය දත්ත වෙත පද්ධතිය ප්‍රවේශ වේ.

The results are generated

ප්‍රතිඵල උත්පාදනය කරනු ලැබේ

The retrieved data is processed and formatted into the desired output.

ලබාගන්නා ලද දත්ත සකසා අපේක්ෂිත ප්‍රතිදානය බවට හැඩගස්වනු ලැබේ.

MySQL and MariaDB

MySQL

- Developed in 1995 by Michael Widenius, Allan Larsson, and David Axmark.
1995 දී Michael Widenius, Allan Larsson සහ David Axmark විසින් සංවර්ධනය කරන ලදී.



- Named after Widenius's first daughter, My. විදේනියස්ගේ මුල්ම දියණිය වන My නමින් එය නම් කරන ලදී.
- It gained global prominence as part of the LAMP stack (Linux, Apache, MySQL, PHP), a foundation for web development.
එය වෙබ් සංවර්ධනය සඳහා පදනමක් වන LAMP stack (Linux, Apache, MySQL, PHP) කොටසක් ලෙස ගෝලීය වශයෙන් ප්‍රසිද්ධියට පත් විය.
- Acquired by Sun Microsystems in 2008 and later by Oracle Corporation in 2010.
2008 දී Sun Microsystems විසින් සහ පසුව 2010 දී Oracle Corporation විසින් අත්පත් කර ගන්නා ලදී.



MariaDB



- Created in 2009 by Michael Widenius as a fork of MySQL, ensuring it remained open-source.
2009 දී මයිකල් විදේනියස් විසින් MySQL හි උපශාඛාවක් ලෙස නිර්මාණය කරනු ලැබූ අතර, එය විවෘත මූලාශ්‍රයක් ලෙස පවතින බවට සහතික කළේය.
- Named after his second daughter, Maria.
ඔහුගේ දෙවන දියණිය වන Maria නමින් එය කරනු ලැබුවේය.
- Designed for backward compatibility with MySQL while introducing performance, scalability, and security improvements.
කාර්ය සාධනය, පරිමාණය සහ ආරක්ෂාව වැඩිදියුණු කිරීම් හඳුන්වා දුන් අතරම MySQL සමඟ පසුගාමී අනුකූලතාව සඳහා නිර්මාණය කර ඇත.
- Supported by the MariaDB Foundation, which promotes its development.
එහි සංවර්ධනය ප්‍රවර්ධනය කරන MariaDB පදනම මගින් මෙයට සහය දැක්වූයේය.

My and Maria with their father, Micheal Widenius.



A Bridge Between Humans and Databases - SQL Interfaces

මිනිසුන් සහ දත්ත සම්ප්‍රදායන් අතර සම්බන්ධතාව SQL අතුරුමුහුණත්

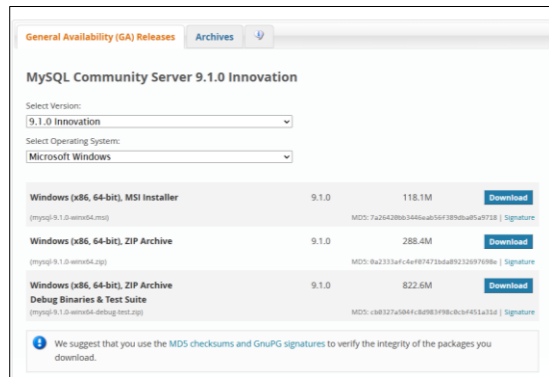
Feature	CLI (Command Line Interface)	GUI (Graphical User Interface)
User interaction පරිශීලක අන්තර්ක්‍රියා	Text-based පාඨ පදනම් වූ Requires manual typing of SQL commands. SQL විධාන අතින් ටයිප් කිරීම අවශ්‍ය වේ.	Visual-based දෘෂ්‍ය පාදක Uses buttons, menus, and forms. බොත්තම්, මෙනු සහ ආකෘති භාවිතා කරයි.
Efficiency කාර්යක්ෂමතාව	Fast for users who are proficient ප්‍රවීණ පරිශීලකයින් සඳහා වේගවත් වේ. Supports scripting and automation. ස්ක්‍රිප්ට්ස් සහ ස්වයංක්‍රීයකරණය සඳහා සහය දක්වයි.	Quick for simple tasks සරල කාර්යයන් සඳහා ඉක්මන් වේ. May be slower for complex operations. සංකීර්ණ මෙහෙයුම් සඳහා මන්දගාමී විය හැක.
Resources usage සම්පත් භාවිතය	Generally, uses fewer system resources. සම්පත් භාවිතය සාමාන්‍යයෙන්, අඩු පද්ධති සම්පත් භාවිතා කරයි.	Can be more resource-intensive due to graphical elements. චිත්‍රක මූලාංග නිසා වැඩි සම්පත් තීව්‍ර විය හැක.
Customization අභිරුචිකරණය	Highly customizable through scripts and commands. ස්ක්‍රිප්ට් සහ විධාන හරහා ඉතා අභිරුචිකරණය කළ හැකිය.	Less customizable limited to available features and options. අඩු අභිරුචිකරණය කළ හැකිය පවතින විශේෂාංග සහ විකල්ප වලට සීමා වේ.
Task Execution Speed කාර්යය ක්‍රියාත්මක කිරීමේ වේගය	Quick for repetitive and complex tasks once mastered. වරක් ප්‍රගුණ කළ පුනරාවර්තන සහ සංකීර්ණ කාර්යයන් සඳහා ඉක්මන් කරන්න.	May be slower for complex tasks but faster for straightforward operations. සංකීර්ණ කාර්යයන් සඳහා මන්දගාමී විය හැකි නමුත් සරල මෙහෙයුම් සඳහා වේගවත් වේ.
User Base පරිශීලක පදනම	Preferred by advanced users and professionals. උසස් පරිශීලකයින් සහ වෘත්තිකයන් විසින් කැමති.	Ideal for beginners and those who prefer visual tools. ආරම්භකයින් සහ දෘෂ්‍ය මෙවලම් කැමති අය සඳහා වඩාත් සුදුසුය.
Popular SQL interfaces ජනප්‍රිය SQL අතුරුමුහුණත්	MySQL (mysql) PostgreSQL (psql) SQL Server (sqlcmd)	MySQL Workbench DBeaver Oracle SQL Developer DataGrip

Installing MySQL only (optional)

Download MySQL from the official site:

නිල වෙබ් අඩවියෙන් MySQL බාගන්න

<https://dev.mysql.com/downloads/mysql/>



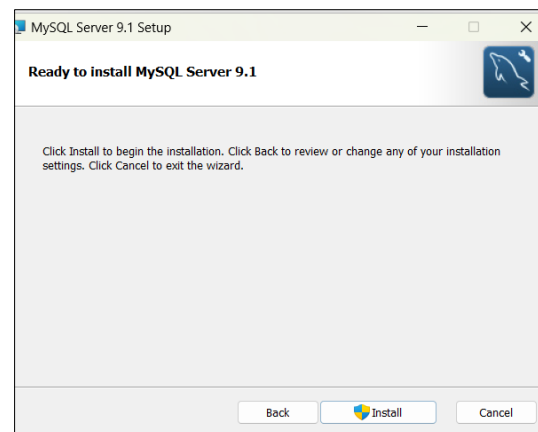
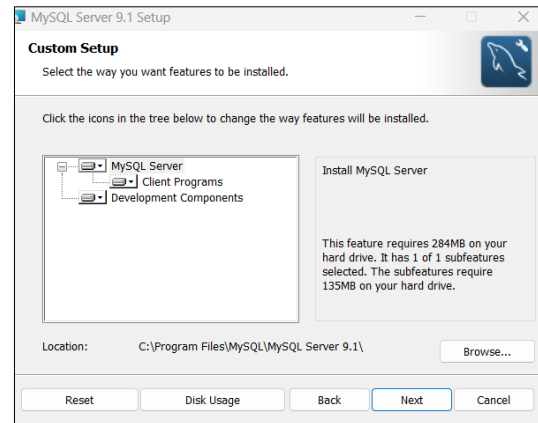
Run the downloaded MSI installer and select 'Custom Installation'.

බාගත කළ MSI ස්ථාපකය ධාවනය කර 'Custom Installation' තෝරන්න.



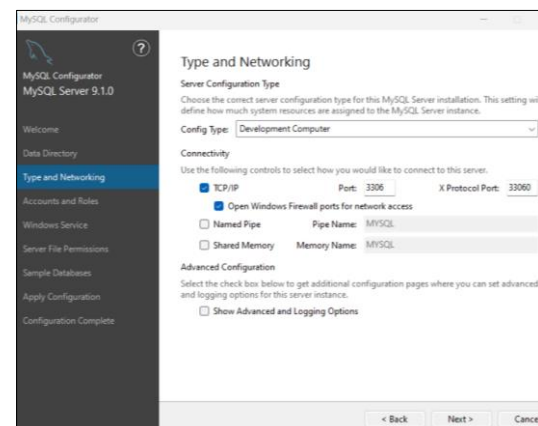
Select the MySQL Server and additional components (like MySQL Workbench).

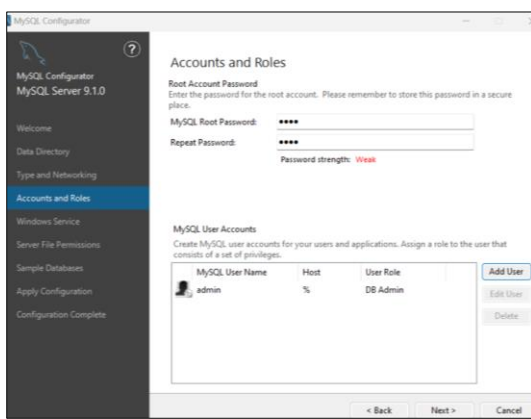
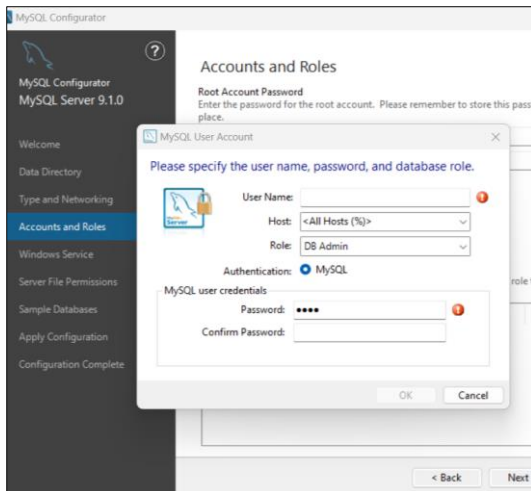
MySQL සේවාදායකය සහ අමතර සංරචක තෝරන්න (MySQL Workbench වැනි).



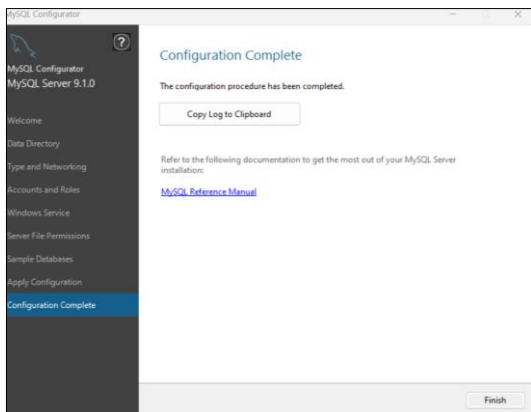
Configure the MySQL Server: Choose a root password and configure the port (default 3306).

MySQL සේවාදායකය වින්‍යාස කරන්න: මූල මුරපදයක් තෝරා port වින්‍යාස කරන්න (පෙරනිමි 3306).





Complete the installation and make a note of the root credentials.
ස්ථාපනය සම්පූර්ණ කර මූල අක්ෂරපත්‍ර සටහන් කරන්න.



Confirm the installation via command prompt by typing:

පහත උපදෙස විධාන ජෙෂ්ට්‍රයෙහි ටයිප් කර ස්ථාපනය තහවුරු කරන්න:

mysql -version

Start the MySQL server via Command Prompt:

විධාන ජෙෂ්ට්‍රය හරහා MySQL සේවාදායකය ආරම්භ කරන්න:

net start MySQL

Launch the MySQL Command-Line Client (CLI) or any other MySQL interface tool like MySQL Workbench.

MySQL විධාන ජෙෂ්ට්‍ර සේවාදායකය හෝ MySQL Workbench වැනි වෙනත් MySQL අතුරුමුහුණත් මෙවලම්ක් දියත් කරන්න.

If you use CLI, in the CLI, connect to the server by typing.

ඔබ CLI භාවිතා කරන්නේ නම් CLI හිදී, පහත උපදෙස ටයිප් කර සේවාදායකයට සම්බන්ධ වන්න

mysql -u your_username -p

Enter your password when prompted.

ඔබෙන් විමසූ විට ඔබගේ මුරපදය ඇතුළත් කරන්න.

Start writing sql codes.

Sql කේත ලිවීම ආරම්භ කරන්න.

Other ways to Install MYSQL

MYSQL ස්ථාපනය කිරීමේ අනිකුත් ක්‍රම

MySQL can be installed on a computer using various software.

MySQL විවිධ මෘදුකාංග භාවිතයෙන් පරිගණකයක ස්ථාපනය කළ හැකිය.

WAMP (Windows)

(Windows, Apache, MySQL, PHP)

WAMP is a popular software bundle specifically designed for Windows operating systems.

WAMP යනු වින්ඩෝස් මෙහෙයුම් පද්ධති සඳහා විශේෂයෙන් නිර්මාණය කර ඇති ජනප්‍රිය මෘදුකාංග කට්ටලයකි.



WAMP provides a convenient and integrated environment for developing and testing web applications on your local machine.

ඔබේ දේශීය යන්ත්‍රයේ වෙබ් යෙදුම් සංවර්ධනය කිරීම සහ පරීක්ෂා කිරීම සඳහා WAMP පහසු සහ ඒකාබද්ධ පරිසරයක් සපයයි.

WAMP Software download Link:

<https://wampserver.aviatechno.net/>



Installation Video guide:

<https://youtu.be/M2at7D-lciw?si=llk5XcS1G1RHHVuzV>



LAMP(LINUX)

(Linux, Apache, MySQL, PHP)



LAMP is another popular open-source software stack for building web applications.

LAMP යනු වෙබ් යෙදුම් තැනීම සඳහා තවත් ජනප්‍රිය විවෘත මූලාශ්‍ර මෘදුකාංග ගොනුවකි.

It's very similar to WAMP but specifically designed for Linux-based systems.

එය WAMP ට බෙහෙවින් සමාන නමුත් ලිනක්ස් මත පදනම් වූ පද්ධති සඳහා විශේෂයෙන් නිර්මාණය කර ඇත.

Just like WAMP, LAMP provides a pre-configured environment for web development. WAMP මෙන්ම, LAMP වෙබ් සංවර්ධනය සඳහා පෙර-විනිසාගත පරිසරයක් සපයයි.

It's widely used due to its reliability, performance, and flexibility.

එහි විශ්වසනීයත්වය, කාර්ය සාධනය සහ නමුණු ලදී බව නිසා එය බහුලව භාවිතා වේ.

LAMP download Link:

<https://ampps.com/lamp/>



Installation Video guide:

<https://youtu.be/iscb52g000M?si=zyvBf1duiNPRpRAA>



MAMP(Mac OS)

(macOs, Apache, MySQL, PHP)



MAMP is a popular software bundle specifically designed for macOS systems.

MAMP යනු macOS පද්ධති සඳහා විශේෂයෙන් නිර්මාණය කර ඇති ජනප්‍රිය මෘදුකාංග ගොනුවකි.

It's similar to WAMP (Windows) and LAMP (Linux) but tailored for Apple users.

එය WAMP(Windows) සහ LAMP(Linux) වලට සමාන නමුත් Apple භාවිතා කරන්නන් සඳහා සකස් කර ඇත.

MAMP download link:

<https://www.mamp.info/en/downloads/>



Installation Video guide:

<https://youtu.be/NqaZ2yqpA4E?si=hhoDU3gOOxIr3QB>



XAMPP

(Cross-Platform, Apache, MySQL, PHP, and Perl)



XAMPP is a popular open-source software package that provides a comprehensive platform for developing and testing web applications locally.

XAMPP යනු දේශීයව වෙබ් යෙදුම් සංවර්ධනය කිරීම සහ පරීක්ෂා කිරීම සඳහා පුළුල් වේදිකාවක් සපයන ජනප්‍රිය විවෘත මූලාශ්‍ර මෘදුකාංග පැකේජයකි.

It's a cross-platform solution, meaning it works on Windows, macOS, and Linux.

එය හරස් වේදිකා විසඳුමකි, එයින් අදහස් වන්නේ එය Windows, macOS, සහ Linux මත ක්‍රියා කරන බවයි.

XAMPP download link:

<https://www.apachefriends.org/download.html>



Installation Video guide:

<https://youtu.be/VCHXCusltql?si=1ZN0QAqRINtg-M4G>



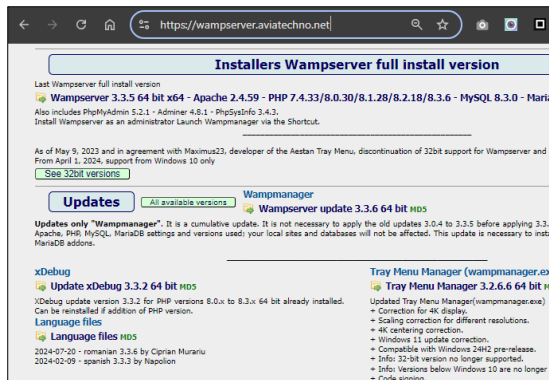
Installing and setting up WAMP

WAMP ස්ථාපනය කිරීම සහ සැකසීම

Download WAMP software from the official website

නිල වෙබ් අඩවියෙන් WAMP මෘදුකාංගය බාගත කරගන්න.

<https://wampserver.aviatechno.net/>

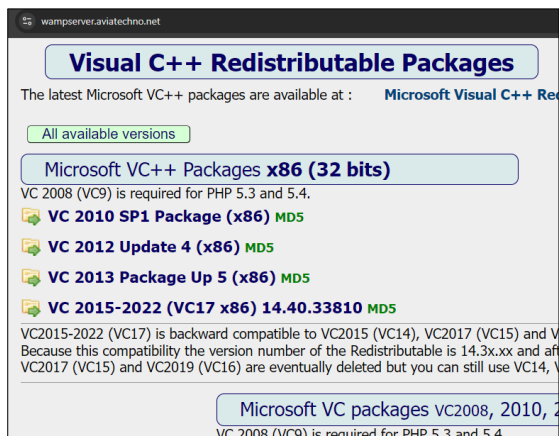


While trying to install WAMP software, you might encounter an error caused by missing **Visual C++ Redistributable Packages**.

WAMP මෘදුකාංගය ස්ථාපනය කිරීමට උත්සාහ කරන අතරතුර, **Visual C++ Redistributable** පැකේජ නොමැති වීම නිසා ඔබට දෝෂයක් ඇති විය හැක.

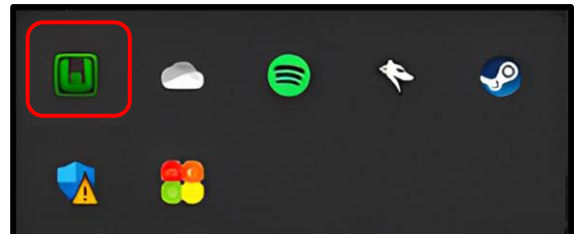
You can also download those missing packages from the WAMP official website.

ඔබට එම නොමැති පැකේජ WAMP නිල වෙබ් අඩවියෙන් බාගත කළ හැකිය.



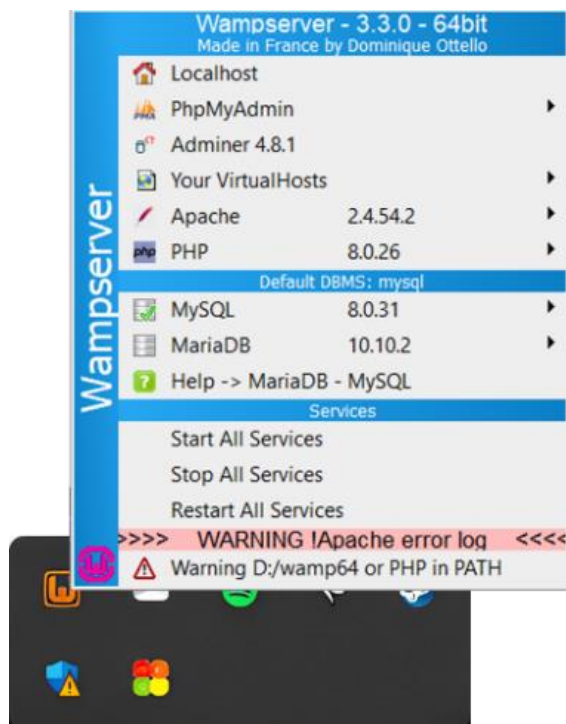
Click the WAMP icon from the desktop or Start Menu. A green icon in the system tray confirms WAMP is running successfully.

ඩෙස්ක්ටොප් එකෙන් හෝ ආරම්භක මෙනුවෙන් WAMP අයිකනය ක්ලික් කරන්න. පද්ධති තැටියේ ඇති කොළ පැහැ නිරූපනය මගින් WAMP සාර්ථකව ක්‍රියාත්මක වන බව තහවුරු කරයි.



Click on the WAMP shortcut.

WAMP shortcut එක මත ක්ලික් කරන්න.

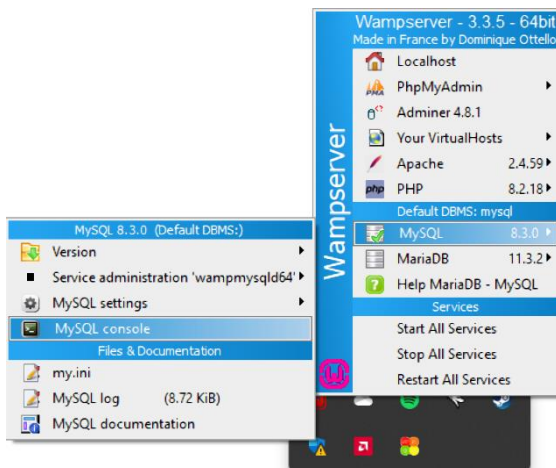


Creating Database Using CLI

CLI අතරමුහුණත භාවිතා කරමින් දත්ත සමුදායක් නිර්මාණය කිරීම

In the menu, navigate to **MySQL → MySQL Console**. Click on "MySQL Console."

මෙහිවේ, **MySQL → MySQL** කොන්සෝලය වෙත සංචාලනය කරන්න. **MySQL Console** මත ක්ලික් කරන්න.



A command prompt (terminal) window will open, asking for the MySQL root password.

MySQL root මුරපදය ඉල්ලා විධාන විමසුම් (command prompt)(පර්යන්ත) කවුළුවක් විවෘත වේ.

If you haven't set a root password during the installation, just press Enter to proceed without entering a password.

ස්ථාපනය අතරතුර ඔබ මූල මුරපදයක් සකසා නොමැති නම්, මුරපදයක් ඇතුළත් නොකර ඉදිරියට යාමට Enter යතුර ඔබන්න.

The MySQL command line interface (CLI) is now open, and MySQL commands can be typed directly into the console to interact with the databases.

MySQL විධාන රේඛා අතරමුහුණත (CLI) දැන් විවෘතව ඇති අතර, දත්ත සමුදායන් සමඟ අන්තර් ක්‍රියා කිරීම සඳහා MySQL විධාන සෘජුවම කොන්සෝලය වෙත ටයිප් කළ හැක.

```
MySQL 8.0 Command Line Cli x + v - □ x
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.37 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

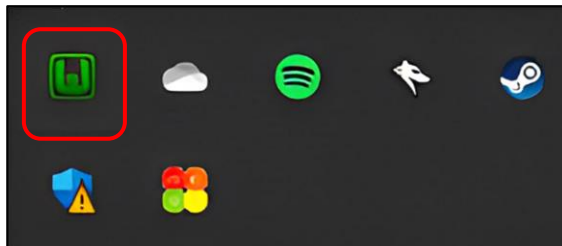
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> |
```

Creating Database Using GUI Interface

GUI අතුරුමුහුණත භාවිතා කරමින් දත්ත සමුදායක් නිර්මාණය කිරීම

Start the WAMP server by clicking the WAMP icon on your desktop. Once it's running, click the WAMP icon in the system tray.

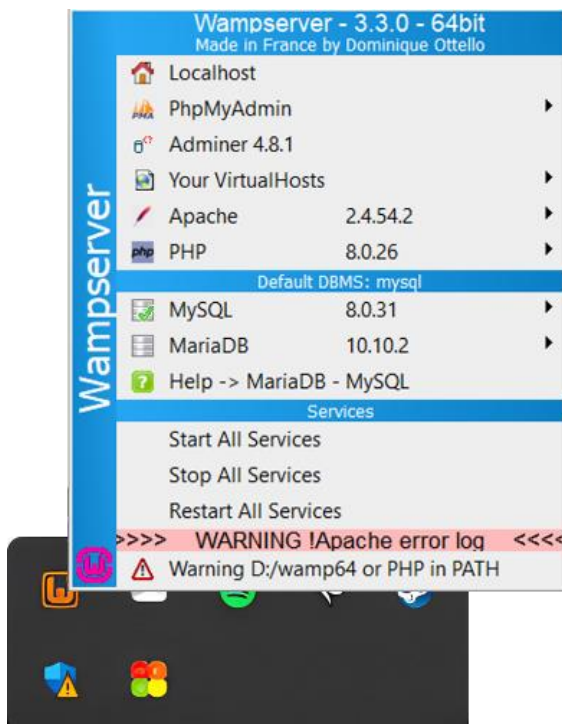
ඔබේ ඩෙස්ක්ටොප් එකේ ඇති WAMP අයිකනය ක්ලික් කිරීමෙන් WAMP සේවාදායකය ආරම්භ කරන්න. එය ක්‍රියාත්මක වූ පසු, පද්ධති තැටියේ ඇති WAMP නිරූපකය ක්ලික් කරන්න.



From the menu that appears, select

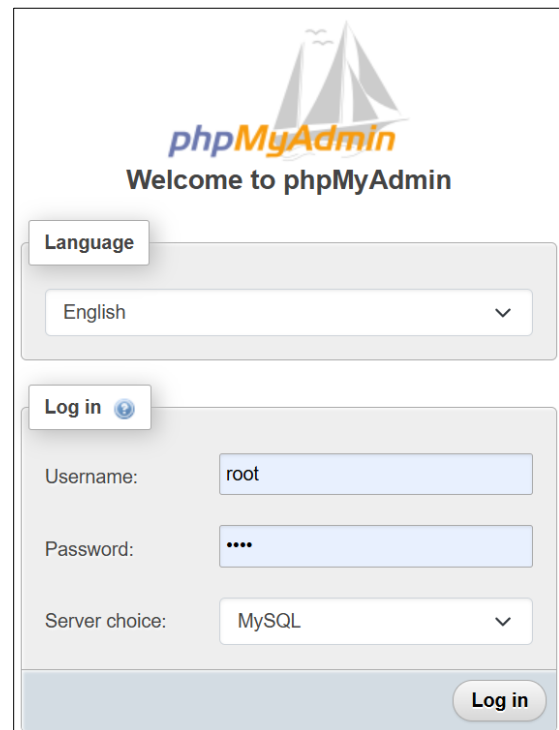
"phpMyAdmin"

දිස්වන මෙනුවෙන්, "phpMyAdmin" තෝරා ගන්න.

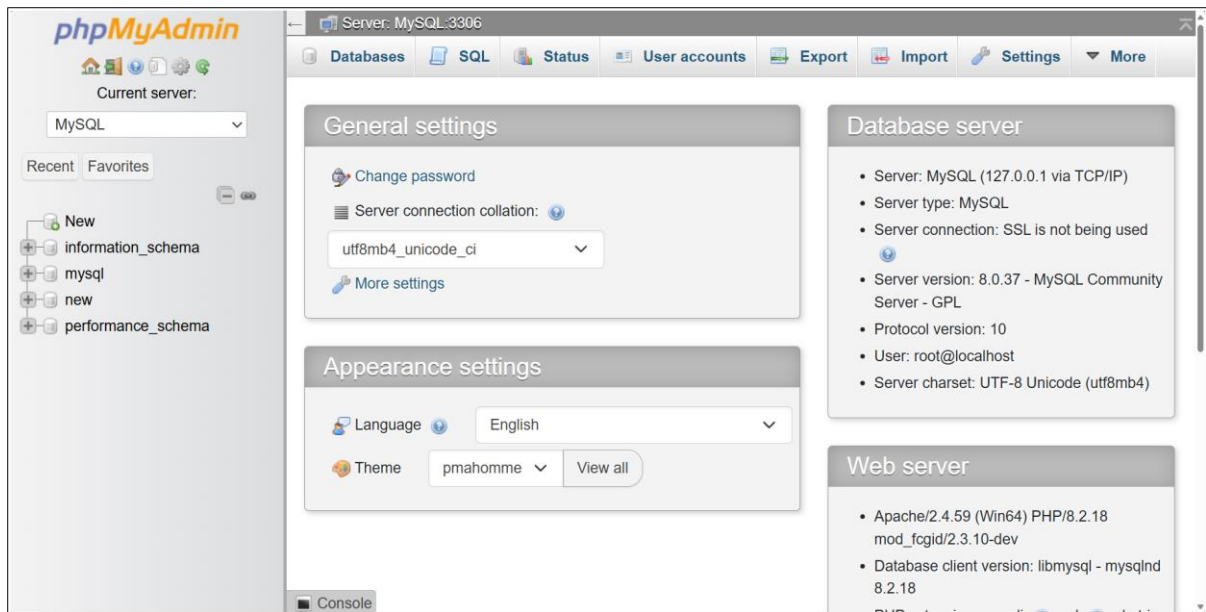


This will open your default web browser and take you to the phpMyAdmin login page.

මෙය ඔබගේ පෙරහිමි වෙබ් බ්‍රව්සරය විවෘත කර ඔබව phpMyAdmin පිවිසුම් පිටුවට ගෙන යනු ඇත.



On the phpMyAdmin login page, enter your MySQL username and password. By default, the username is "root" and the password is usually blank unless you set one during WAMP installation. Click the "Log in" button to log in. phpMyAdmin පිවිසුම් පිටුවේ, ඔබේ MySQL පරිශීලක නාමය සහ මුරපදය ඇතුළත් කරන්න. පෙරහිමියෙන්, පරිශීලක නාමය "root" වන අතර, ඔබ WAMP ස්ථාපනය කිරීමේදී එකක් සකසන්නේ නම් මිස මුරපදය සාමාන්‍යයෙන් හිස්ව පවතී. පුරනය වීමට "Log in" බොත්තම ක්ලික් කරන්න.

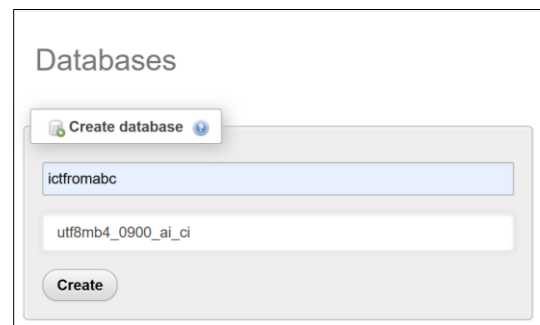


Once logged in, you will be on the phpMyAdmin main page. Click on the "Database" tab at the top. In the "Create database" field, enter the desired database name

ලොග් වූ පසු, ඔබ phpMyAdmin ප්‍රධාන පිටුවෙහි සිටිනු ඇත. ඉහළින් ඇති "Database" ටැබය මත ක්ලික් කරන්න. "Create database" ක්ෂේත්‍රය තුළ, අවශ්‍ය දත්ත සමුදා නාමය ඇතුළත් කරන්න.

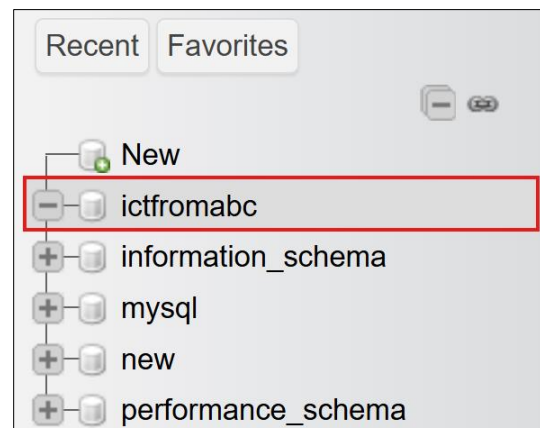
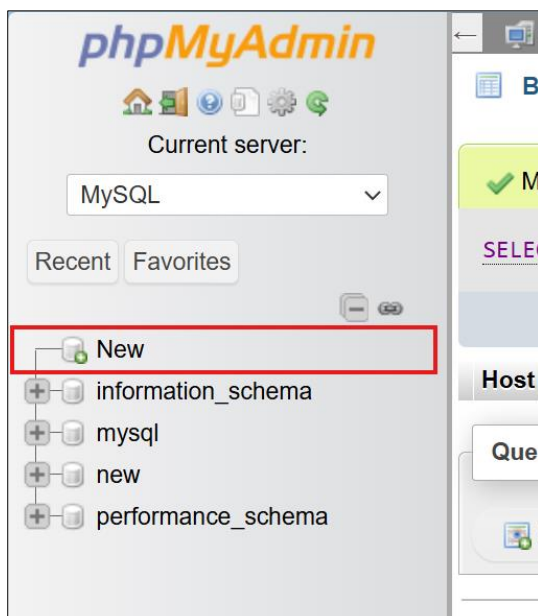
After entering the name, click the "Create" button.

නම ඇතුළත් කිරීමෙන් පසු, "Create" බොත්තම ක්ලික් කරන්න.



New database will now be listed on the left side of the phpMyAdmin interface.

නව දත්ත සමුදාය දැන් phpMyAdmin අතුරුමුහුණතේ වම් පැත්තේ ලැයිස්තුගත කෙරේ.



1. SHOW DATABASES;

Shows available databases.
පවතින දත්ත සමුදායන් පෙන්වයි.

SHOW SCHEMAS; can also be used.
SHOW SCHEMAS; ද භාවිත කළ හැකිය.

2. CREATE DATABASE ;

new database name

Creates a new database.
නව දත්ත සමුදායක් නිර්මාණය කරයි.

CREATE SCHEMA; can also be used.
CREATE SCHEMA; ද භාවිත කළ හැකිය.

3. USE ;

existing database name

Selects an existing database.
පවතින දත්ත සමුදායක් තෝරාගනී.

CONNECT database_name; can also be used.
CONNECT database_name; ද භාවිත කළ හැකිය.

4. SHOW TABLES;

Shows all available tables in a database.
දත්ත සමුදායක පවතින සියලුම වගු පෙන්වයි.

NOTE

First, we should select the database.
ප්‍රථමයෙන් දත්ත සමුදාය තෝරාගත යුතුය.

5. DROP DATABASE ;

existing database name

Used to delete an existing database.
පවතින දත්ත සමුදායක් මකා දැමීමට භාවිත කරයි.

DROP SCHEMA; can also be used.
DROP SCHEMA; ද භාවිත කළ හැකිය.

6. Creating a new table නව වගුවක් සෑදීම

new table name

CREATE TABLE  (

  datatype

 );

column name

comma

Example:

Student

RegNo	Name	Address	Age

Relational Schema

සම්බන්ධතා පටිපාටිය

Student (RegNo, Name, Address, Age)

```
CREATE TABLE Student (  
    RegNo INT(4),  
    Name VARCHAR(20),  
    Address VARCHAR(40),  
    age VARCHAR(3)  
);
```

7. DROP TABLE ;

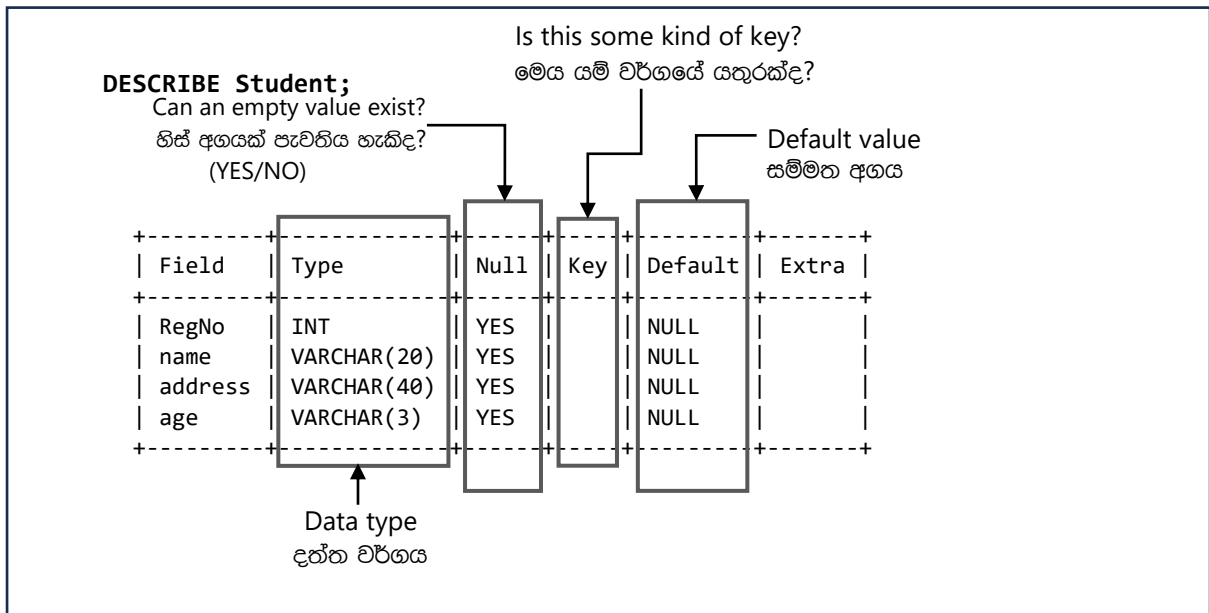
existing table name

Deletes an existing table from a database.
දත්ත සමුදායකින් පවතින වගුවක් මකා දමයි.

8. DESCRIBE ;

existing table name

Describes the structure of an existing table.
පවතින වගුවක ව්‍යුහය පිළිබඳ විස්තර සපයයි.



Data types in SQL language

SQL භාෂාවේ දත්ත වර්ග

Data types are labels assigned to table columns that specify the kind of data they can store. They act as containers for specific type of data.

දත්ත වර්ග යනු වගුවල තීරු වලට පවරා ඇති ලේඛල වන අතර ඒවා මගින් තීරු තුළ ගබඩා කළ හැකි දත්ත වර්ගය නියම කරයි. ඒවා නිශ්චිත දත්ත වර්ගයක් සඳහා බහාලුම් ලෙස ක්‍රියා කරයි.

Why are they important?

ඒවා වැදගත් වන්නේ ඇයි

Ensures data consistency and accuracy.
දත්ත අනුකූලතාව සහ නිරවද්‍යතාවය සහතික කිරීමට.

Optimizes database performance by using appropriate storage methods.
සුදුසු ගබඩා ක්‍රම භාවිතා කරමින් දත්ත සම්ප්‍රදායේ ක්‍රියාකාරීත්වය ප්‍රශස්ත කිරීමට.

Helps prevent invalid data from being entered.
වලංගු නොවන දත්ත ඇතුළත් කිරීම වැළැක්වීමට උපකාරී වීම

Reduces runtime errors caused by mismatched data.
නොගැලපෙන දත්ත නිසා ඇතිවන ධාවන දෝෂ අඩු කරයි.

Incorrect data type usage can lead to truncation or rounding.
වැරදි දත්ත වර්ග භාවිතය ඒවා මැකී යාමට හෝ වැට්ටීමට හේතු විය හැක

Data types act as the first layer of defense against malicious inputs.
අනිශ්චිත ආදාන වලට වරෙහිව ආරක්ෂාවේ පළමු ස්ථරය ලෙස දත්ත වර්ග ක්‍රියා කරයි

Determines which operations (arithmetic, concatenation) are valid for the data.
දත්ත සඳහා වලංගු වන්නේ කුමන මෙහෙයුම් (අංක ගණිතමය, concatenation) ද යන්න තීරණය කරයි.

There are 3 main types of data types in SQL language.

SQL භාෂාවේ ප්‍රධාන දත්ත වර්ග ආකාර 3ක් පවතී.

- String Data Types
- Numeric Data Types
- Date and Time Data Types

String Data Types

SQL string data types store alphanumeric data, including characters and text. They come in various forms to efficiently manage different types of text data.

SQL string දත්ත වර්ග අක්ෂර සහ පාඨ ඇතුළත් අක්ෂරාංක දත්ත ගබඩා කරයි. විවිධ වර්ගයේ පෙළ දත්ත කාර්යක්ෂමව කළමනාකරණය කිරීම සඳහා ඒවා විවිධ ආකාරවලින් පැමිණේ.

Numeric Data Types

SQL numeric data types store numeric values in formats like integers, fixed-point, and floating-point numbers. They ensure the precision and range needed for mathematical and financial calculations.

SQL සංඛ්‍යාත්මක දත්ත නිඛිල, ස්ථාවර ස්ථාවර ලක්ෂ්‍යය සහ ඉපිලෙන ලක්ෂ්‍ය සංඛ්‍යා වැනි ආකෘතිවල සංඛ්‍යාත්මක අගයන් ගබඩා කරයි. ඔවුන් ගණිතමය හා මූල්‍ය ගණනය කිරීම් සඳහා අවශ්‍ය නිරවද්‍යතාවය සහ පරාසය සහතික කරයි.

Date and Time Data Types

SQL date and time data types are used to store date and time values in various formats. These data types are essential for managing and manipulating temporal data, such as dates, times, timestamps, and intervals.

SQL දින සහ වේලා දත්ත වර්ග විවිධ ආකෘතිවලින් දිනය සහ කාල අගයන් ගබඩා කිරීමට භාවිතා කරයි. දින, වේලාවන්, වේලා මුද්‍රා සහ කාල පරතරයන් වැනි තාවකාලික දත්ත කළමනාකරණය කිරීම සහ හැසිරවීම සඳහා මෙම දත්ත වර්ග අත්‍යවශ්‍ය වේ

STRINGS

Data Type	Description	Maximum Length
CHAR	To represent letters, numbers or any character with a specified number of characters නිශ්චිත අනුලක්ෂණ ප්‍රමාණයක් ඇති අකුරු, ඉලක්කම් හෝ ඕනෑම අනුලක්ෂණයක් නිරූපනය සඳහා	n characters
VARCHAR	To include any attribute with an unspecified length නිශ්චිත නොවන දිගක් සහිත ඕනෑම අනුලක්ෂණයක් ඇතුලත් කිරීම සඳහා	n characters
TEXT	To enter a standard text. සම්මත පාඨයක් ඇතුලත් කිරීම සඳහා.	65,535 bytes (64 KB)
TINYTEXT	Very small text string. ඉතා කුඩා පාඨ string.	255 bytes
MEDIUMTEXT	Medium-length text string. මධ්‍යම දිග පාඨ string.	16,777,215 bytes (16 MB)
LONGTEXT	Large text string. විශාල පාඨ string.	4,294,967,295 bytes (4 GB)
BINARY	Fixed-length binary data. ස්ථාවර දිග ද්විමය දත්ත.	n bytes
VARBINARY	Variable-length binary data. චලන දිග ද්විමය දත්ත.	n bytes

NUMERIC

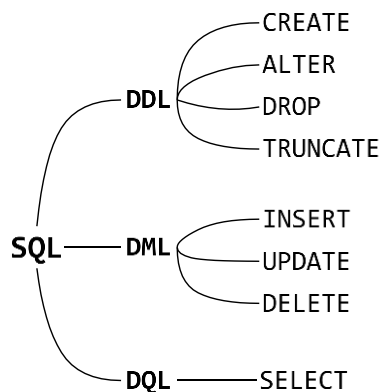
Data Type	Description	Storage	Range
INT	Representation of positive or negative integers ධන හෝ සෘණ නිඛිල සංඛ්‍යා නිරූපණය	4 bytes	-2,147,483,648 to 2,147,483,647
TINYINT	Very small integer ඉතා කුඩා පූර්ණ සංඛ්‍යා	1 byte	0 to 255 (unsigned) or -128 to 127 (signed)
SMALLINT	Small integer කුඩා පූර්ණ සංඛ්‍යා	2 bytes	-32,768 to 32,767
MEDIUMINT	Medium-sized integer මධ්‍යම ප්‍රමාණයේ පූර්ණ සංඛ්‍යා	3 bytes	-8,388,608 to 8,388,607
BIGINT	Large integer විශාල නිඛිල	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
FLOAT	Representation of positive or negative floating point numbers ධන හෝ සෘණ දශම සංඛ්‍යා නිරූපණය	4 bytes	Approximate range: $\sim 1.5 \times 10^{-45}$ to $\sim 3.4 \times 10^{38}$
BIT	Binary values (0 or 1) ද්විමය අගයන් 0 හෝ 1	Variable	1 or 0

DATE & TIME

Data Type	Description	Format	Example
DATE	Stores date values දින අගයන් ගබඩා කරයි	YYYY-MM-DD	2024-07-26
TIME	Stores time values වේලා අගයන් ගබඩා කරයි	HH:MM:SS	13:45:30
DATETIME	Stores combined date and time values ඒකාබද්ධ දින සහ වේලා අගයන් ගබඩා කරයි	YYYY-MM-DD HH:MM:SS	2024-07-26 13:45:30
TIMESTAMP	Stores date and time values, includes time zone information දින සහ කාල අගයන් ගබඩා කරයි, වේලා කලාප තොරතුරු ඇතුළත් වේ	YYYY-MM-DD HH:MM:SS	2024-07-26 13:45:30
YEAR	Stores year values අවුරුදු අගයන් ගබඩා කරයි	YYYY	2024

Classification of SQL codes

SQL කේත වර්ගීකරණය



i) Data Definition Language (DDL)

දත්ත අර්ථකථන භාෂාව

Used to define and manage database schema and structure.
දත්ත සමුදා ක්‍රමය සහ ව්‍යුහය නිර්වචනය කිරීමට සහ කළමනාකරණය කිරීමට භාවිතා කරයි.

CREATE DROP
ALTER TRUNCATE

ii) Data Manipulation Language (DML)

දත්ත හැසුරුම් භාෂාව

Used for managing data within database objects.
දත්ත සමුදා වස්තූන් තුළ දත්ත කළමනාකරණය කිරීම සඳහා භාවිතා කරයි.

INSERT UPDATE
DELETE CALL

iii) Data Query Language (DQL)

දත්ත විමසුම් භාෂාව

Used to query the database.
දත්ත සමුදාය විමසීමට භාවිතා කරයි.

SELECT

iv) Data Control Language (DCL)

දත්ත පාලන භාෂාව

Used to control access to data within the database.
දත්ත සමුදාය තුළ දත්ත වෙත ප්‍රවේශය පාලනය කිරීමට භාවිතා කරයි.

GRANT REVOKE

v) Transaction Control Language (TCL)

ගනුදෙනු පාලන භාෂාව

Manage transactions within the database.
දත්ත ගබඩාව තුළ ගනුදෙනු කළමනාකරණය කරයි

COMMIT ROLLBACK
SAVEPOINT SET TRANSACTION

Naming styles in SQL

SQL හි නම් කිරීමේ විලාස

- Two database names within the same database management system cannot be the same.
එකම දත්ත සමුදා කළමනාකරණ පද්ධතිය තුළ ඇති දත්ත සමුදා නාම දෙකක් සමාන විය නොහැක.
- Two table names within the same database cannot be the same.
එකම දත්ත සමුදාය තුළ ඇති වගු දෙකක නම් සමාන විය නොහැක.
- Two column names within the same table cannot be the same.
එකම වගුව තුළ ඇති තීරු දෙකක නම් සමාන විය නොහැක.
- Two table names can be the same if they are in different databases.
වගු දෙකක් එකිනෙකට වෙනස් දත්ත සමුදායන්හි පවතී නම් ඒවායේ නම් සමාන විය හැකිය.
- Column names can be the same in different tables, even within the same database (not recommended).
එකම දත්ත සමුදාය තුළ වුවද, වෙනස් වගු වල ඇති තීරු දෙකක නම් සමාන විය හැකිය.

Table and Column names; වගු සහ තීරු නම්වල;

- can contain letters, digits, and underscores.
අකුරු, ඉලක්කම් සහ යටි ඉරි අඩංගු විය හැකිය.
- cannot contain special characters (such as spaces, @, #, -, etc.).
විශේෂ අක්ෂර අඩංගු විය නොහැක (හිස්තැන්, ?, *, -, ආදිය).
- must start with a letter or an underscore.
අකුරකින් හෝ යටි ඉරි ලකුණකින් ආරම්භ විය යුතුය.
- cannot start with a digit.
ඉලක්කමකින් ආරම්භ කළ නොහැක.

ALTER TABLE Command

ALTER TABLE විධානය

Used to modify the structure of an existing table.

දැනට පවතින වගුවක ව්‍යුහය වෙනස් කිරීමට භාවිතා කරයි.

1. Adding a New Column to a Table

වගුවකට නව ක්ෂේත්‍රයක් එක් කිරීම

ALTER TABLE  existing table name
ADD COLUMN    ;
column name data type constraints

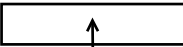
2. Removing an existing field from a table.

වගුවකින් පවතින ක්ෂේත්‍රයක් ඉවත් කිරීම.

ALTER TABLE  existing table name
DROP COLUMN  ;
column name

3. Changing the name of an existing field

පවතින ක්ෂේත්‍රයක නම වෙනස් කිරීම

ALTER TABLE  existing table name
RENAME COLUMN  TO  ;
previous column name new column name

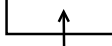
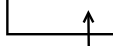
ALTER TABLE MODIFY

Used to change the definition of an existing column.

පවතින තීරුවක අර්ථ දැක්වීම් වෙනස් කිරීමට භාවිතා කරයි.

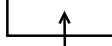
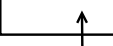
a) Can change the data type of a column.

තීරුවක දත්ත වර්ගය වෙනස් කළ හැකිය.

existing table name
ALTER TABLE 
MODIFY COLUMN   ;
column name new data type

b) Can change the data size of a column.

තීරුවක දත්ත ප්‍රමාණයේ විශාලත්වය වෙනස් කළ හැකිය.

existing table name
ALTER TABLE 
MODIFY COLUMN   ;
column name data type with new size

c) Can add certain constraints to a column.

තීරුවකට යම් මට්ටමකට සම්බන්ධ ව්‍යවස්ථාපිත කළ හැකිය.

NOT NULL

Changes a column so it can no longer be empty.

තීරුවක් තවදුරටත් හිස්ව පැවතිය නොහැකි වන පරිදි වෙනස් කරයි.

Ex.

```
ALTER TABLE Employees MODIFY  
COLUMN last_name VARCHAR(50) NOT  
NULL;
```

DEFAULT

Sets a fallback value if no data is entered.

දත්ත ඇතුළත් කර නොමැති විට ඒ සඳහා වන ගැලපෙන අගයක් සකසයි.

Ex.

```
ALTER TABLE Employees MODIFY  
COLUMN status VARCHAR(20) DEFAULT  
'Active';
```

UNIQUE

Ensures no two rows have the same value in this column.

අදාල තීරුවේ ජේලි දෙකක අනුපිටපත් වූ අගයන් නොමැති බව සහතික කරයි.

Ex.

```
ALTER TABLE Employees MODIFY
COLUMN email VARCHAR(100) UNIQUE;
```

AUTO_INCREMENT

Makes the number count up automatically when you add new rows.

නව ජේලි එකතු කරන විට සංඛ්‍යාව ස්වයංක්‍රීයව වැඩි වන ලෙස සකසයි.

Ex.

```
ALTER TABLE Employees MODIFY
COLUMN employee_id INT
AUTO_INCREMENT;
```

CHECK

Adds a logic rule that data must follow.

දත්ත විසින් අනුගමනය කළ යුතු තාර්කික රීතියක් එකතු කරයි.

Ex.

```
ALTER TABLE Employees MODIFY
COLUMN salary INT CHECK (salary >
0);
```

PRIMARY KEY

Makes this column the main identifier for the table (only if one doesn't exist yet).

මෙම අදාල තීරුව වගුව සඳහා වූ ප්‍රධාන හඳුන්වනය බවට පත් කරයි (දැනටමත් හඳුන්වනයක් නොපවතී නම් පමණි).

Ex.

```
ALTER TABLE Employees MODIFY
COLUMN employee_id INT PRIMARY
KEY;
```

Examples for ALTER TABLE

ALTER TABLE සඳහා උදාහරණ

Example table

Field	Type	Null	Key	Default	Extra
RegNo	int	YES		NULL	
Name	varchar(20)	YES		NULL	
Address	varchar(40)	YES		NULL	
Age	varchar(3)	YES		NULL	

ALTER TABLE student

ADD COLUMN Phone VARCHAR(10) NOT NULL;

Describe student;

Field	Type	Null	Key	Default	Extra
RegNo	int	YES		NULL	
Name	varchar(20)	YES		NULL	
Address	varchar(40)	YES		NULL	
Age	varchar(3)	YES		NULL	
Phone	varchar(10)	NO		NULL	

```
ALTER TABLE student
RENAME COLUMN Phone TO Contact;
```

```
Describe student;
```

Field	Type	Null	Key	Default	Extra
RegNo	int	YES		NULL	
Name	varchar(20)	YES		NULL	
Address	varchar(40)	YES		NULL	
Age	varchar(3)	YES		NULL	
Contact	varchar(10)	NO		NULL	

```
ALTER TABLE student
DROP COLUMN Contact;
```

```
Describe student;
```

Field	Type	Null	Key	Default	Extra
RegNo	int	YES		NULL	
Name	varchar(20)	YES		NULL	
Address	varchar(40)	YES		NULL	
Age	varchar(3)	YES		NULL	

```
ALTER TABLE student
MODIFY COLUMN Age INT;
```

```
Describe student;
```

Field	Type	Null	Key	Default	Extra
RegNo	int	YES		NULL	
Name	varchar(20)	YES		NULL	
Address	varchar(40)	YES		NULL	
Age	int	YES		NULL	

```
ALTER TABLE student
MODIFY COLUMN Name VARCHAR(30);
```

```
Describe student;
```

Field	Type	Null	Key	Default	Extra
RegNo	int	YES		NULL	
Name	varchar(30)	YES		NULL	
Address	varchar(40)	YES		NULL	
Age	int	YES		NULL	

INSERT INTO Statement

Used to add new rows of data to a table.
වගුවකට නව දත්ත පේළි එක් කිරීමට භාවිතා කරයි.

Can be used to insert data into specific columns or all columns of a table, insert multiple rows, and even insert data from another table.

වගුවක නිශ්චිත තීරුවලට හෝ සියලුම තීරුවලට දත්ත ඇතුළු කිරීමට, පේළි කිහිපයක් ඇතුළු කිරීමට සහ වෙනත් වගුවකින් දත්ත ඇතුළු කිරීමට පවා භාවිත කළ හැක.

Inserting a Single Record

තනි වාර්තාවක් ඇතුළත් කිරීම

a) All columns

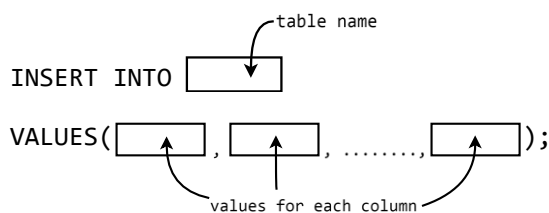
සියලුම තීරු

Only the data should be defined in the order in which the column names are present in the database table.

දත්ත සමුදාය වගුවේ තීරු නාම පවතින අනුපිළිවෙලට දත්ත පමණක් අර්ථ දැක්වීම සිදු කර යුතුය.

Here values should be applied to each column of the table.

මෙහිදී වගුවේ සෑම තීරුවකටම අගයන් යෙදිය යුතුය.

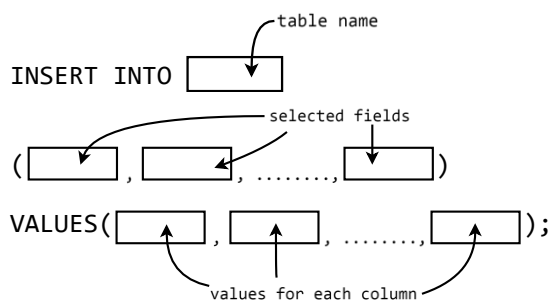


b) Specific columns

නිශ්චිත තීරු

Here, only a few columns of data can be defined.

මෙහිදී තීරු කිහිපයකට පමණක් දත්ත අර්ථ දැක්වීම සිදු කළ හැක.



Inserting Multiple Records

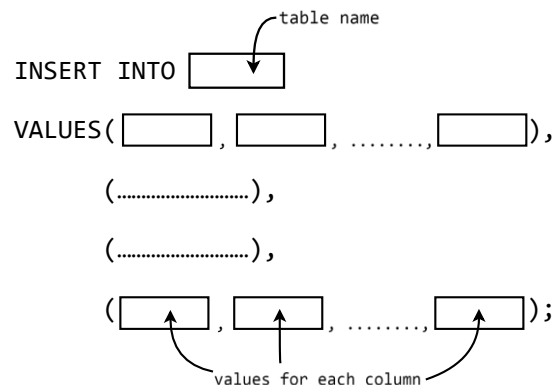
වාර්තා කිහිපයකට ඇතුළත් කිරීම

Can insert multiple records into a table in a single INSERT INTO statement by separating each set of values with a comma.

එක් එක් අගයන් කට්ටලයක් කොමාවකින් වෙන් කිරීමෙන් තනි INSERT INTO ප්‍රකාශයක් තුළ වගුවකට වාර්තා කිහිපයක් ඇතුළත් කළ හැක.

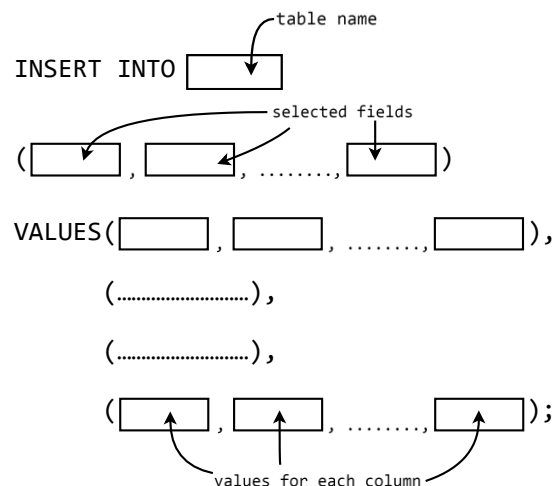
a) All columns

සියලුම තීරු



b) Specific columns

නිශ්චිත තීරු



Examples

Reference Table:

```
CREATE TABLE Employee (  
  RegNo INT,  
  Name VARCHAR(15),  
  Address VARCHAR(20),  
  TelNo VARCHAR(10)  
);
```

1) Single Record to all columns

එකී වාර්තාවක් සියලුම තීරු වලට

```
INSERT INTO Employee  
VALUES (1, 'Akon',  
'Colombo', '0711111111');
```

RESULT

```
SELECT * FROM Employee;  
+-----+-----+-----+-----+  
| RegNo | Name | Address | TelNo |  
+-----+-----+-----+-----+  
| 1 | Akon | Colombo | 0711111111 |  
+-----+-----+-----+-----+
```

2) Single Record to specific columns

එකී වාර්තාවක් නිශ්චිත තීරු වලට

```
INSERT INTO Employee  
(RegNo, Name, Address)  
VALUES (2, 'Bkon', 'Gampaha');
```

RESULT

```
SELECT * FROM Employee;  
+-----+-----+-----+-----+  
| RegNo | Name | Address | TelNo |  
+-----+-----+-----+-----+  
| 1 | Akon | Colombo | 0711111111 |  
| 2 | Bkon | Gampaha | NULL |  
+-----+-----+-----+-----+
```

3) Multiple Records to all columns

වාර්තා කිහිපයක් සියලුම තීරු වලට

```
INSERT INTO Employee  
VALUES (3, 'Ckon', 'Kandy',  
'0733333333'),  
(4, 'Dkon', 'Kurunagala',  
'0744444444');
```

RESULT

```
SELECT * FROM Employee;  
+-----+-----+-----+-----+  
| RegNo | Name | Address | TelNo |  
+-----+-----+-----+-----+  
| 1 | Akon | Colombo | 0711111111 |  
| 2 | Bkon | Gampaha | NULL |  
| 3 | Ckon | Kandy | 0733333333 |  
| 4 | Dkon | Kurunagala | 0744444444 |  
+-----+-----+-----+-----+
```

4) Multiple Records to specific columns

වාර්තා කිහිපයක් නිශ්චිත තීරු වලට

```
INSERT INTO Employee  
(RegNo, Name, TelNo)  
VALUES (5, 'Ekon', '0755555555'),  
(6, 'Fkon', '0766666666');
```

RESULT

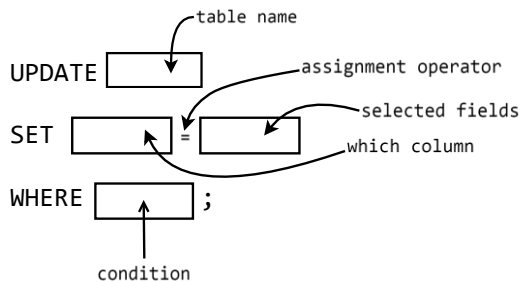
```
SELECT * FROM Employee;  
+-----+-----+-----+-----+  
| RegNo | Name | Address | TelNo |  
+-----+-----+-----+-----+  
| 1 | Akon | Colombo | 0711111111 |  
| 2 | Bkon | Gampaha | NULL |  
| 3 | Ckon | Kandy | 0733333333 |  
| 4 | Dkon | Kurunagala | 0744444444 |  
| 5 | Ekon | NULL | 0755555555 |  
| 6 | Fkon | NULL | 0766666666 |  
+-----+-----+-----+-----+
```

UPDATE Statement

Used to modify existing records in a table.
වගුවක පවතින වාර්තා වෙනස් කිරීමට භාවිතා කරයි.

You can update one or more columns for one or more rows by specifying the conditions that must be met for a row to be updated.

පේළියක් යාවත්කාලීන කිරීම සඳහා සපුරාලිය යුතු කොන්දේසි සඳහන් කිරීමෙන් ඔබට පේළි එකක් හෝ කිහිපයක් සඳහා තීරු එකක් හෝ කිහිපයක් යාවත්කාලීන කළ හැකිය.



a) One column → one record

එක් ක්ෂේත්‍රයක් → එක් රෙකෝඩයක්

```
UPDATE Employee
SET Address='Galle'
WHERE RegNo=5;
```

RESULT

```
SELECT * FROM Employee;
```

RegNo	Name	Address	TelNo
1	Akon	Colombo	0711111111
2	Bkon	Gampaha	NULL
3	Ckon	Kandy	0733333333
4	Dkon	Kurunagala	0744444444
5	Ekon	Galle	0755555555
6	Fkon	NULL	0766666666

b) One column → many records

එක් ක්ෂේත්‍රයක් → රෙකෝඩ කීපයක්

```
UPDATE Employee
SET TelNo='0700000000'
WHERE RegNo<3;
```

RESULT

```
SELECT * FROM Employee;
```

RegNo	Name	Address	TelNo
1	Akon	Colombo	0700000000
2	Bkon	Gampaha	0700000000
3	Ckon	Kandy	0733333333
4	Dkon	Kurunagala	0744444444
5	Ekon	Galle	0755555555
6	Fkon	NULL	0766666666

c) Many columns → one record

ක්ෂේත්‍ර කීපයක් → එක් රෙකෝඩයක්

```
UPDATE Employee
SET Address='Matara',
TelNo='0777777777'
WHERE RegNo=6;
```

RESULT

```
SELECT * FROM Employee;
```

RegNo	Name	Address	TelNo
1	Akon	Colombo	0700000000
2	Bkon	Gampaha	0700000000
3	Ckon	Kandy	0733333333
4	Dkon	Kurunagala	0744444444
5	Ekon	Galle	0755555555
6	Fkon	Matara	0777777777

NOTE:

Using the **UPDATE** command without the **WHERE** condition will update all records of the table.

WHERE කොන්දේසිය නොමැතිව **UPDATE** විධානය භාවිතයෙන් වගුවේ සියලුම රෙකෝඩ් යාවත්කාලීන වීම සිදුවේ.

```
UPDATE Employee
SET Address='Colombo';
```

RESULT

```
SELECT * FROM Employee;
```

RegNo	Name	Address	TelNo
1	Akon	Colombo	0700000000
2	Bkon	Colombo	0700000000
3	Ckon	Colombo	0733333333
4	Dkon	Colombo	0744444444
5	Ekon	Colombo	0755555555
6	Fkon	Colombo	0777777777

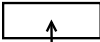
DELETE Statement

Used to remove one or more records from a table.

වගුවකින් රෙකෝඩ් එකක් හෝ කිහිපයක් ඉවත් කිරීමට භාවිතා කරයි.

The rows to be deleted are specified using a condition in the WHERE clause.

මකා දැමිය යුතු පේළි WHERE වගන්තියේ කොන්දේසියක් භාවිතයෙන් නියම කර ඇත.

DELETE FROM 
WHERE  ;

↑
condition

↑
table name

a) Single record එක් රෙකෝඩයක්

```
DELETE FROM Employee  
WHERE RegNo=4;
```

RESULT

```
SELECT * FROM Employee;
```

RegNo	Name	Address	TelNo
1	Akon	Colombo	0700000000
2	Bkon	Colombo	0700000000
3	Ckon	Colombo	0733333333
5	Ekon	Colombo	0755555555
6	Fkon	Colombo	0777777777

b) Multiple records රෙකෝඩ් කීපයක්

```
DELETE FROM Employee  
WHERE RegNo>3;
```

RESULT

```
SELECT * FROM Employee;
```

RegNo	Name	Address	TelNo
1	Akon	Colombo	0700000000
2	Bkon	Colombo	0700000000
3	Ckon	Colombo	0733333333

c) All records සියලු රෙකෝඩ්

```
DELETE FROM Employee;
```

RESULT

```
SELECT * FROM Employee;  
Empty set (0.00 sec)
```

TRUNCATE TABLE table_name; can also be used.

TRUNCATE TABLE table_name; ද භාවිතා කළ හැකිය.

SELECT Statement

Used to retrieve data from one or more tables in a database.

දත්ත සම්ප්‍රදායක ඇති වගු එකකින් හෝ වැඩි ගණනකින් දත්ත ලබා ගැනීමට භාවිතා කරයි.

The results are returned in the form of a result set, which is displayed in a table format.

ප්‍රතිඵල, ප්‍රතිඵල-කට්ටලයක ආකාරයෙන් ආපසු ලබා දෙනු ලැබේ, එය වගු ආකෘතියකින් පෙන්වනු ලැබේ.

SELECT ← which column
FROM ← which table
WHERE ;
 ↑
 which records

The WHERE clause must include either a Comparison operator or a logical operator.

WHERE වගන්තිය සඳහා සැසඳුම් කාරක හෝ තාර්කික කාරක ඇතුළත් කළ යුතුය.

Employee

RegNo	Name	Address	AGE
1	Akon	Kandy	30
2	Bkon	Nugegoda	28
3	Ckon	Colombo	35
4	Dkon	Nugegoda	40

a) All columns → all records

සියලුම ක්ෂේත්‍ර → සියලුම රෙකෝඩ්

SELECT * FROM Employee;

RESULT

RegNo	Name	Address	AGE
1	Akon	Kandy	30
2	Bkon	Nugegoda	28
3	Ckon	Colombo	35
4	Dkon	Nugegoda	40

b) Multiple columns → all records

ක්ෂේත්‍ර කීපයක් → සියලුම රෙකෝඩ්

SELECT Name, Age FROM Employee;

RESULT

Name	Age
Akon	30
Bkon	28
Ckon	35
Dkon	40

c) Single column → all records

එක් ක්ෂේත්‍රයක් → සියලුම රෙකෝඩ්

SELECT Name FROM Employee;

RESULT

Name
Akon
Bkon
Ckon
Dkon

d) Single column → multiple records

එක් ක්ෂේත්‍රයක් → රෙකෝඩ් කීපයක්

SELECT Name FROM Employee
WHERE RegNo>2;

RESULT

Name
Ckon
Dkon

e) Single column → single record

එක් ක්ෂේත්‍රයක් → එක් රෙකෝඩයක්

SELECT Name FROM Employee
WHERE RegNo=1;

RESULT

Name
Akon

f) Multiple columns → multiple records

ක්ෂේත්‍ර කීපයක් → රෙකෝඩ් කීපයක්

```
SELECT Name, Age FROM Employee
WHERE RegNo>2;
```

RESULT

Name	Age
Ckon	35
Dkon	40

g) Multiple columns → single record

ක්ෂේත්‍ර කීපයක් → එක් රෙකෝඩයක්

```
SELECT Name, Age FROM Employee
WHERE RegNo=2;
```

RESULT

Name	Age
Bkon	28

h) All columns → multiple records

සියලුම ක්ෂේත්‍ර → රෙකෝඩ් කීපයක්

```
SELECT * FROM Employee
WHERE RegNo>2;
```

RESULT

RegNo	Name	Address	AGE
3	Ckon	Colombo	35
4	Dkon	Nugegoda	40

Comparison Operators

සැසඳුම් කාරක

Used in the WHERE clause to compare two expressions and filter records based on the result of that comparison.

WHERE වගන්තියේ ප්‍රකාශන දෙකක් සංසන්දනය කිරීමට සහ එම සැසඳීමේ ප්‍රතිඵලය මත පදනම්ව රෙකෝඩ් පෙරහන් කිරීමට භාවිතා කරයි.

Essential for creating conditions that retrieve, update, or delete specific data in a database.

දත්ත සමූදායක නිශ්චිත දත්ත ලබා ගැනීම, යාවත්කාලීන කිරීම හෝ මකා දැමීම සඳහා කොන්දේසි නිර්මාණය කිරීම සඳහා අත්‍යවශ්‍ය වේ.

Use cases of comparison operator

සැසඳුම් කාරකයන්හි භාවිත

- Filter data to retrieve specific records that meet a given condition.
දී ඇති කොන්දේසියක් සපුරාලන නිශ්චිත වාර්තා ලබා ගැනීමට දත්ත තෝරා ගැනීමට.
- To implement conditional logic in queries.
විමසුම්වල කොන්දේසි සහිත තර්ක ක්‍රියාත්මක කිරීමට.
- To identify records that have missing or undefined values.
නැතිවූ හෝ නිර්වචනය නොකළ අගයන් ඇති වාර්තා හඳුනා ගැනීමට.
- To find records that match an exact value.
නිශ්චිත අගයකට ගැලපෙන වාර්තා සොයා ගැනීමට.

Notation අංකනය	Operation මෙහෙයවනය
=	equal සමාන
<>	not equal අසමාන
!=	not equal අසමාන
>	greater than විශාල
<	less than කුඩා
>=	greater than or equal විශාල හෝ සමාන
<=	less than or equal කුඩා හෝ සමාන
BETWEEN	Between a certain range නිශ්චිත පරාසයක් අතර BETWEEN ____ AND ____
IN	Saturate at least one value from a given set of values දී ඇති අගයන් සමූහයක් අතරින් එක් අගයක් හෝ තෘප්ත කර ගත IN(____,____,____)
LIKE	search for a character pattern අනුලක්ෂණ රටාවක් සෙවීමට

REFERENCE TABLE

Employee

RegNo	Name	Address	Age
001	AKON	KANDY	30
002	BELLA	NUGEGODA	28
003	CAMERON	COLOMBO	35
004	DIANA	NUGEGODA	40

= Comparison Operator

= සැසඳුම් කාරකය

Employees whose address is NUGEGODA
ලිපිනය නුගේගොඩ වන සේවකයන්

```
SELECT * FROM Employee
WHERE Address = "NUGEGODA";
```

RegNo	Name	Address	Age
2	BELLA	NUGEGODA	28
4	DIANA	NUGEGODA	40

<> or != Comparison Operator

<> or != සැසඳුම් කාරකය

Employees whose address is not NUGEGODA
ලිපිනය නුගේගොඩ නොවන සේවකයන්

```
SELECT * FROM Employee
WHERE Address <> "NUGEGODA";
```

OR

```
SELECT * FROM Employee
WHERE Address != "NUGEGODA";
```

RegNo	Name	Address	Age
1	AKON	KANDY	30
3	CAMERON	COLOMBO	35

> Comparison Operator

> සැසඳුම් කාරකය

Employees above 30 years of age

වයස 30 ට වඩා වැඩි සේවකයන්

```
SELECT * FROM Employee
WHERE Age > 30;
```

RegNo	Name	Address	Age
003	CAMERON	COLOMBO	35
004	DIANA	NUGEGODA	40

< Comparison Operator

< සැසඳුම් කාරකය

Employees below 30 years of age

වයස 30 ට වඩා අඩු සේවකයන්

```
SELECT * FROM Employee
WHERE Age < 30;
```

RegNo	Name	Address	Age
002	BELLA	NUGEGODA	28

>= Comparison Operator

>= සැසඳුම් කාරකය

Employees aged 30 years or more

වයස 30ට සමාන හෝ වැඩි සේවකයන්

```
SELECT * FROM Employee
WHERE Age >= 30;
```

RegNo	Name	Address	Age
001	AKON	KANDY	30
003	CAMERON	COLOMBO	35
004	DIANA	NUGEGODA	40

<= Comparison Operator

<= සැසඳුම් කාරකය

Employees aged 30 years or less

වයස 30ට සමාන හෝ අඩු සේවකයන්

```
SELECT * FROM Employee
WHERE Age <= 30;
```

RegNo	Name	Address	Age
001	AKON	KANDY	30
002	BELLA	NUGEGODA	28

BETWEEN Comparison Operator

BETWEEN සැසඳුම් කාරකය

Employees between 30 and 40 years of age

වයස අවුරුදු 30ත් 40ත් අතර සේවකයන්

```
SELECT * FROM Employee
WHERE Age BETWEEN 30 AND 40;
```

RegNo	Name	Address	Age
001	AKON	KANDY	30
003	CAMERON	COLOMBO	35
004	DIANA	NUGEGODA	40

IN Comparison Operator

IN සැසඳුම් කාරකය

Employees whose address is NUGEGODA or KANDY

ලිපිනය නුගේගොඩ හෝ මහනුවර වන සේවකයන්

```
SELECT * FROM Employee
WHERE Address
IN("NUGEGODA", "KANDY");
```

RegNo	Name	Address	Age
001	AKON	KANDY	30
002	BELLA	NUGEGODA	28
004	DIANA	NUGEGODA	40

SQL Wildcards

Used in conjunction with the LIKE operator in the WHERE clause to search for patterns within text fields.

පාඩ් ක්ෂේත්‍ර තුළ රටා සෙවීමට WHERE වගන්තියේ ඇති LIKE ක්‍රියාකරු සමඟ ඒකාබද්ධව භාවිතා වේ.

- % Matches zero or more characters
අනුලක්ෂණ 0ක් හෝ කිහිපයක් ගලපයි
- _ Matches a single character
අනුලක්ෂණ 1ක් ගලපයි

1. Beginning with a specific letter

නිශ්චිත අකුරකින් ආරම්භ වන

Details of employees whose name starts with letter B

B අකුරින් නම පටන් ගන්නා සේවකයින්ගේ විස්තර

Employee

RegNo	Name	Address	Age
001	AKON	KANDY	30
002	BELLA	NUGEGODA	28
003	CAMERON	COLOMBO	35
004	BENNY	GALLE	40

```
SELECT * FROM Employee
WHERE Name LIKE "B%";
```

RegNo	Name	Address	Age
002	BELLA	NUGEGODA	28
004	BENNY	GALLE	40

2. Ending with a specific letter

නිශ්චිත අකුරකින් අවසන් වන

Details of employees whose name ends with letter A

A අකුරින් නම අවසන් වන සේවකයින්ගේ විස්තර

Employee

RegNo	Name	Address	Age
001	AKON	KANDY	30
002	BELLA	NUGEGODA	28
003	CAMERON	COLOMBO	35
004	BENNY	GALLE	40

```
SELECT * FROM Employee
WHERE Name LIKE "%A";
```

RegNo	Name	Address	Age
002	BELLA	NUGEGODA	28

REFER TO THE FOLLOWING TABLE FOR 3 AND 4 SUBTOPICS IN THE NEXT PAGE

මීලඟ පිටුවේ ඇති 3 සහ 4 මාතෘකා සඳහා පහත වගුව වෙත යොමු වන්න.

Employee

RegNo	Name	Address	Age
001	SHANE	KANDY	30
002	HANNAH	NUGEGODA	28
003	CHARLIE	COLOMBO	35
004	MICHAEL	GALLE	40
005	SARAH	MATHARA	32

3. A specific letter or letters present at any position

ඕනෑම ස්ථානයක පවතින නිශ්චිත අකුරක් හෝ අකුරු

Details of employees where HA exists at any place in the name

නමේ ඕනෑම ස්ථානයක HA පවතින සේවකයින්ගේ විස්තර

```
SELECT * FROM Employee
WHERE Name LIKE "%HA%";
```

RegNo	Name	Address	Age
001	SHANE	KANDY	30
002	HANNAH	NUGEGODA	28
003	CHARLIE	COLOMBO	35
004	MICHAEL	GALLE	40

4. There is a specific letter in the second position

දෙවැනි ස්ථානයේ නිශ්චිත අකුරක් පවතින

Details of employees whose names have letter A at the second position

නමේ දෙවැනි අකුර A වන සේවකයින්ගේ විස්තර

```
SELECT * FROM Employee
WHERE Name LIKE "_A%";
```

RegNo	Name	Address	Age
002	HANNAH	NUGEGODA	28
005	SARAH	MATHARA	32

5. Starting with a specified letter and ending with a specified letter

නිශ්චිත අකුරකින් ආරම්භ වී නිශ්චිත අකුරකින් අවසන් වන

Details of employees whose names start with letter H and end with letter A

H අකුරෙන් ආරම්භ වන A අකුරෙන් අවසන් වන නම් ඇති සේවකයින්ගේ විස්තර

Employee

RegNo	Name	Address	Age
001	HANNAH	COLOMBO	25
002	HERA	GALLE	30
003	HENRY	KANDY	28
004	HLENA	NUGEGODA	35

```
SELECT * FROM Employee
WHERE Name LIKE "H%A";
```

RegNo	Name	Address	Age
002	HERA	GALLE	30
004	HLENA	NUGEGODA	35

LOGICAL OPERATORS

තාර්කික මෙහෙයවන

Used to combine two or more conditions in SQL queries.

SQL විමසුම්වල කොන්දේසි දෙකක් හෝ වැඩි ගණනක් ඒකාබද්ධ කිරීමට භාවිතා කරයි.

Reference Table

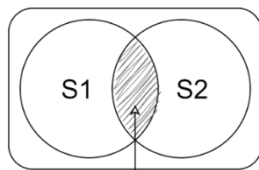
Employee

RegNo	Name	Address	age
001	AKON	KANDY	35
002	BKON	GAMPAHA	40
003	CKON	KANDY	32

1) AND OPERATOR

Will return TRUE only if both conditions are met.

TRUE ලෙස ලබා දෙන්නේ කොන්දේසි දෙකම සපුරා ඇත්නම් පමණි.



S1 AND S2

```
SELECT * FROM Employee
WHERE Address = 'KANDY' AND age > 30;
```

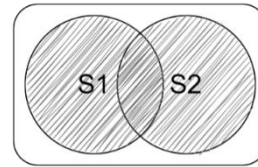
RESULT

RegNo	Name	Address	age
001	AKON	KANDY	35
003	CKON	KANDY	32

2) OR OPERATOR

Will return TRUE only if at least one of the conditions is met.

TRUE ආපසු ලබා දෙන්නේ අවම වශයෙන් එක් කොන්දේසියක්වත් සපුරා ඇත්නම් පමණි.



S1 OR S2

```
SELECT * from Employee
WHERE age > 33 OR Address = 'GAMPAHA';
```

RESULT

RegNo	Name	Address	age
001	AKON	KANDY	35
002	BKON	GAMPAHA	40

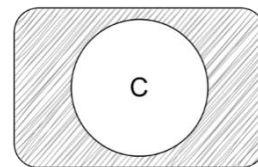
3) NOT OPERATOR

Used to reverse the result of a condition.

කොන්දේසියක ප්‍රතිඵලය ආපසු හැරවීමට භාවිතා කරයි.

Selects rows that do not meet the specified criteria.

නිශ්චිත කොන්දේසියක් සපුරා නොමැති පේළි තෝරයි.



NOT C

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

```
SELECT * FROM customers
WHERE NOT Address = 'KANDY';
```

RESULT

RegNo	Name	Address	age
002	BKON	GAMPAHA	40

ORDER BY keyword

SELECT [] ← which column
FROM [] ← which table
ORDER BY [] ASC / DESC ;
 ↑
 which column

Reference Table

Employee

RegNo	Name	Address	Age
1	HELEN	GALLE	33
2	CHARLIE	KANDY	27
3	ALICE	COLOMBO	28
4	EDWARD	COLOMBO	35

1. Ascending (ASC)

ආරෝහණ

EXAMPLE

Arrange all details of employees in ascending order of their name.

සේවකයින්ගේ සියළු විස්තර ඔවුන්ගේ නමේ ආරෝහණ පිළිවෙලට සැකසීම.

```
SELECT * FROM Employee
ORDER BY Name;
```

OR

```
SELECT * FROM Employee
ORDER BY Name ASC;
```

The default is ASC for the selected column.
 තෝරාගන්න ලද කීරුව සඳහා පෙරනිමිය ASC වේ

RESULT

RegNo	Name	Address	Age
3	ALICE	COLOMBO	28
2	CHARLIE	KANDY	27
4	EDWARD	COLOMBO	35
1	HELEN	GALLE	33

2. Descending (DESC)

අවරෝහණ

EXAMPLE

Arrange all employee details in descending order of name.

සේවකයින්ගේ සියළු විස්තර ඔවුන්ගේ නමේ අවරෝහණ පිළිවෙලට සකසීම.

```
SELECT * FROM Employee
ORDER BY Name DESC;
```

RESULT

RegNo	Name	Address	Age
1	HELEN	GALLE	33
4	EDWARD	COLOMBO	35
2	CHARLIE	KANDY	27
3	ALICE	COLOMBO	28

3. ORDER BY Several Columns

තීරු කිහිපයකින් පෙළගැස්වීම

When used with multiple columns, SQL sorts the data first by the first column. If there are same values, it then sorts by the next column, and continues in the same way.

කිරු කිහිපයක් සමඟ භාවිතා කරන විට, SQL මඟින් දත්ත පළමුව, ප්‍රථම කිරුව අනුව වර්ග කරයි. එහි එකම අගයන් තිබේ නම්, එය ඊළඟ කිරුව අනුව වර්ග කරන අතර මෙය දිගටම සිදු කරයි.

SELECT ← which column

FROM ← which table

ORDER BY ,

ASC / DESC ;

 which column

EXAMPLE

Sorting all details of employees, names of employees in ascending order and then in descending order by salary

සේවකයින්ගේ සියලුම විස්තර, සේවකයින්ගේ නමෙහි
ආරෝහණ පිළිවෙලින් සහ පසුව වැටුපෙහි අවරෝහණ
පිළිවෙලට සැකසීම

Employee

RegNo	Name	Address	Salary
1	ALICE	COLOMBO	30000
2	BOB	COLOMBO	25000
3	ALICE	COLOMBO	32000

```
SELECT * FROM Employee
ORDER BY Name ASC, Salary DESC;
```

RESULT

RegNo	Name	Address	Salary
3	ALICE	COLOMBO	32000
1	ALICE	COLOMBO	30000
2	BOB	COLOMBO	25000

GROUP BY statement

Group rows with the same values into summary rows.

එකම අගයන් සහිත පේළි සාරාංශ පේළි වලට කාණ්ඩ කරයි.

often used with functions to group the result-set by one or more columns.

බොහෝ විට ශ්‍රිත සමඟ භාවිතා කරනුයේ ප්‍රතිඵල සකසන ලද තීරු එකකින් හෝ කිහිපයකින් කාණ්ඩ කිරීමටයි.

```
SELECT  ← which column
FROM  ← table name
WHERE  ← condition
GROUP BY  ;
           ↑
         which column
```

EXAMPLE

Getting the total cost that the employees had spent.

සේවකයින් වියදම් කළ මුළු පිරිවැය ලබා ගැනීම.

Expenses

CostID	RegNo	Name	Address	Cost
1	1	Akon	Colombo	55000
2	2	Bkon	Nugegoda	60000
3	1	Akon	Colombo	10000
4	3	Ckon	Gampaha	10000
5	2	Bkon	Nugegoda	5000

With conditions

කොන්දේසි සහිතව

```
SELECT Name, SUM(Cost)
FROM Expenses
WHERE Cost > 5000
GROUP BY Name;
```

RESULT

Name	SUM(Cost)
Akon	65000
Bkon	60000
Ckon	10000

Without conditions

කොන්දේසි නොමැතිව

```
SELECT Name, SUM(Cost)
FROM Expenses
GROUP BY Name;
```

RESULT

Name	SUM(Cost)
Akon	65000
Bkon	65000
Ckon	10000

HAVING clause

Was added to SQL because the WHERE keyword cannot be used with aggregate functions.

WHERE මූල පදය සමස්ත ශ්‍රිත සමග භාවිතා කළ නොහැකි නිසා මෙය SQL වෙත එකතු කර ඇත.

HAVING filters **groups** (after GROUP BY), while WHERE filters **rows** (before grouping).

HAVING මගින් කණ්ඩායම් (GROUP BY ට පසුව) තෝරන අතර, WHERE මගින් පේළි (කණ්ඩායම් කිරීමට පෙර) තෝරනු ලබයි.

SELECT ← which column
FROM ← table name
WHERE ← condition
GROUP BY ← which column
HAVING ;
 ↑
 condition

EXAMPLE

Expenses

CostID	RegNo	Name	Address	Cost
1	1	Akon	Colombo	55000
2	2	Bkon	Nugegoda	60000
3	1	Akon	Colombo	10000
4	3	Ckon	Gampaha	10000
5	2	Bkon	Nugegoda	5000

With conditions (WHERE)

කොන්දේසි සහිතව (WHERE)

```
SELECT Name, SUM(Cost)
FROM Expenses
WHERE Cost > 10000
GROUP BY Name
HAVING SUM(Cost) > 10000;
```

RESULT

Name	SUM(Cost)
Akon	55000
Bkon	60000

First it filters rows where Cost is greater than 10,000.

පළමුවෙන් Cost එක 10,000 ට වඩා වැඩි පේළි තෝරාගනී.

It then groups the remaining rows by Name.

ඉන්පසු තෝරාගත් පේළි නම අනුව කාණ්ඩ කරයි.

For each name, it adds up the Cost values using SUM.

සෑම නමක් සඳහාම, SUM භාවිතයෙන් Cost අගයන් එකතු කරයි.

Finally, the HAVING clause keeps only those groups whose total cost exceeds 10,000.

අවසාන වශයෙන්, HAVING වගන්තිය මගින් තෝරා ගන්නේ එක 10,000 ඉක්මවන කාණ්ඩ පමණි.

Without conditions (WHERE)

කොන්දේසි නොමැතිව (WHERE)

```
SELECT Name, SUM(Cost)
FROM Expenses
GROUP BY Name
HAVING SUM(Cost) > 10000;
```

RESULT

Name	SUM(Cost)
Akon	65000
Bkon	65000

DISTINCT statement

Used to return only different values.

වෙනස් අගයන් පමණක් ආපසු ලබා දීමට භාවිත කරයි.

```
SELECT DISTINCT  ← which column
FROM  ;
```

table name

EXAMPLE

Expenses

CostID	RegNo	Name	Address	Cost
1	1	Akon	Colombo	55000
2	2	Bkon	Nugegoda	60000
3	1	Akon	Colombo	10000
4	3	Ckon	Gampaha	10000
5	2	Bkon	Nugegoda	5000

```
SELECT DISTINCT Name
FROM Expenses;
```

RESULT

Name
Akon
Bkon
Ckon

SQL FUNCTION

SQL ශ්‍රිත

SQL functions are regarded as powerful tools that allow data within a database to be manipulated and transformed. They can be used to perform various operations.

SQL ශ්‍රිතයන් දත්ත සම්ප්‍රදායක් තුළ දත්ත හැසිරවීමට සහ පරිවර්තනය කිරීමට ඉඩ සලසන බලවත් මෙවලම් ලෙස සැලකේ. විවිධ මෙහෙයුම් සිදු කිරීමට ඒවා භාවිත කළ හැකිය.

These functions are essential tools for performing complex queries and data analysis in SQL. Each function serves a specific purpose and can be used in combination to achieve desired results.

මෙම ශ්‍රිතයන් SQL හි සංකීර්ණ විමසුම් සහ දත්ත විශ්ලේෂණය සිදු කිරීම සඳහා අත්‍යවශ්‍ය මෙවලම් වේ. සෑම කාර්යයක්ම නිශ්චිත අරමුණක් ඉටු කරන අතර අපේක්ෂිත ප්‍රතිඵල ලබා ගැනීම සඳහා ඒකාබද්ධව භාවිත කළ හැක.

Operation මෙහෙයුම	Examples උදාහරණ
Mathematical calculations ගණිතමය ගණනය කිරීම්	addition, subtraction, multiplication, division එකතු කිරීම, අඩු කිරීම, ගුණ කිරීම, බෙදීම
Data manipulation දත්ත හැසිරවීම	converting data types, extracting substrings දත්ත වර්ග පරිවර්තනය කිරීම, substring උපුටා ගැනීම
Statistical analysis සංඛ්‍යානමය විශ්ලේෂණය	calculating averages, maximums, minimums සාමාන්‍යය, උපරිම, අවම ගණනය කිරීම
Date and time operations දිනය සහ වේලාව මෙහෙයුම්	getting the current date, calculating differences වත්මන් දිනය ලබා ගැනීම, වෙනස්කම් ගණනය කිරීම

Below are some commonly used SQL functions.

පහත දැක්වෙන්නේ බහුලව භාවිතා වන SQL ශ්‍රිත කිහිපයකි.

EXAMPLE TABLE

Employee

RegNo	Name	Address	Salary
1	AKON	KANDY	50000
2	BKON	COLOMBO	60000
3	CKON	GALLE	55000

1. SUM()

The SUM() function calculates the total of a numeric column.

SUM() ශ්‍රිතය සංඛ්‍යාත්මක තීරුවක එකතුව ගණනය කරයි.

```
SELECT SUM(Salary)
FROM Employee;
```

SUM(Salary)
165000

2. AVG()

The AVG() function calculates the average value of a numeric column.

AVG() ශ්‍රිතය සංඛ්‍යාත්මක තීරුවක සාමාන්‍ය අගය ගණනය කරයි.

```
SELECT AVG(Salary)
FROM employee;
```

AVG(Salary)
55000.0000

3. COUNT()

The COUNT() function counts the number of rows that match a specified condition.

COUNT() ශ්‍රිතය නිශ්චිත කොන්දේසියකට ගැලපෙන පේළි ගණන ගණන් කරයි.

Number of employees with a salary greater than 50000

50000 ට වැඩි වැටුපක් ලැබූ සේවකයන් සංඛ්‍යාව

```
SELECT COUNT(*)
FROM Employee
WHERE Salary > 50000;
```

COUNT(*)
2

1. MAX()

Returns the highest value in a column.

තීරුවක ඉහළම අගය ලබා දෙයි.

```
SELECT MAX(Salary)
FROM Employee;
```

MAX(Salary)
60000

2. MIN()

Returns the lowest value in a column.

තීරුවක අඩුම අගය ලබා දෙයි.

```
SELECT MIN(Salary)
FROM employee;
```

MIN(Salary)
50000

Additional functions

අතිරේක ශ්‍රිත

Function	Definition	Example code
CONCAT()	Joins two or more strings together. string දෙකක් හෝ කිහිපයක් එකට සම්බන්ධ කිරීමට.	SELECT CONCAT(fName, '', lName) FROM students;
UPPER()	Converts a string to uppercase. string එකක් uppercase ට පරිවර්තනය කිරීමට.	SELECT UPPER(f_name) FROM students;
LOWER()	Converts a string to lowercase. string එකක් lowercase ට පරිවර්තනය කිරීමට.	SELECT LOWER(f_name) FROM students;
SUBSTRING()	Extracts a portion of a string. string එකක කොටසක් උපුටා ගැනීමට.	SELECT SUBSTRING(f_name, 1, 3) FROM employees;
LENGTH()	Returns the length of a string. string එකක දිග ආපසු ලබා ගැනීමට.	SELECT LENGTH(f_name) FROM employees;

Aliases

Aliases are temporary names assigned to a table or a column to simplify queries, improve readability, or make the output more user-friendly.

Aliases යනු විමසුම් සරල කිරීමට, කියවීමේ හැකියාව වැඩිදියුණු කිරීමට හෝ ප්‍රතිදානය වඩාත් පරිශීලක-හිතකාමී කිරීමට වගුවකට හෝ තීරුවකට පවරා ඇති තාවකාලික නම් වේ.

Column Aliases

```
SELECT first_name AS Name
FROM employees;
```

Table Aliases

```
SELECT e.first_name,d.department_name
FROM employees AS e
JOIN departments AS d
ON e.department_id = d.department_id;
```

Aliases with Functions

Aliases are commonly used with functions to make the output more readable by assigning a meaningful name to the computed result.
ගණනය කළ ප්‍රතිඵලයට අර්ථවත් නමක් ලබා දීමෙන් ප්‍රතිදානය වඩාත් කියවිය හැකි බවට පත් කිරීම සඳහා ශ්‍රිතයන් සමඟ බහුලව Aliases භාවිතා වේ.

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

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

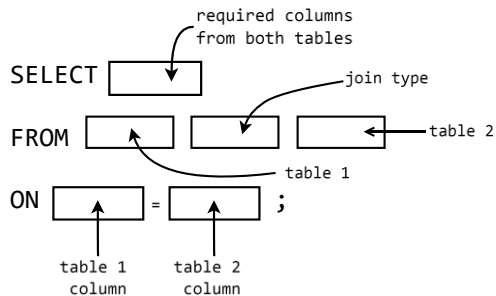
```
SELECT MAX(salary) AS HighestSalary
FROM employees;
```

```
SELECT department_id, AVG(salary) AS
AvgSalary FROM employees GROUP BY
department_id;
```

SQL JOINS

Used to combine rows from two or more tables based on a related column between them.

වගු දෙකකින් හෝ කිහිපයකින් එම වගු සියල්ලට සම්බන්ධ තීරුවක් භාවිතයෙන් ජෝයින් ඒකාබද්ධ කිරීමට භාවිතා කරයි.



For the explanations of joins, we use the following tables.

Joins පැහැදිලි කිරීම සඳහා, අපි පහත වගු භාවිතා කරමු.

Student

StID	StName	SpID
1	Akon	100
2	Bkon	101
3	Ckon	102

Sport

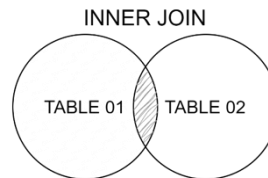
SportID	SportName
101	Chess
102	Carrom
103	Football

Types of Joins

INNER JOIN

Selects records that have matching values in both tables.

වගු දෙකෙහිම ගැළපෙන අගයන් ඇති වාර්තා තෝරා ගනී.



```
SELECT *
FROM Student
INNER JOIN sport
ON Student.SpID = Sport.SportID;
```

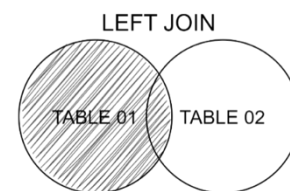
RESULT

StID	StName	SpID	SportID	SportName
2	Bkon	101	101	Chess
3	Ckon	102	102	Carrom

LEFT JOIN

Returns all records from the left table, and the matching records from the right table.

වම්පස වගුවේ සියලුම වාර්තා සහ දකුණුපස වගුව හා ගැළපෙන වාර්තා ලබා දෙයි.



```
SELECT *
FROM Student
LEFT JOIN sport
ON Student.SpID = Sport.SportID;
```

RESULT

StID	StName	SpID	SportID	SportName
1	Akon	100	NULL	NULL
2	Bkon	101	101	Chess
3	Ckon	102	102	Carrom

NOTE

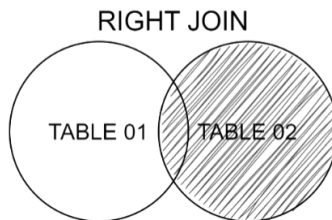
If there is no matching record found, the result will be generated as NULL in right hand side.

ගැළපෙන වාර්තාවක් හමු නොවූයේ නම්, දකුණු පස ප්‍රතිඵලය NULL ලෙස පෙන්වනු ලබයි.

RIGHT JOIN

Returns all records from the right table, and the matching records from the left table.

දකුණුපස වගුවේ සියලුම වාර්තා සහ වම් වගුව හා ගැලපෙන වාර්තා ලබා දෙයි.



```
SELECT *
FROM Student
RIGHT JOIN sport
ON Student.SpID = Sport.SportID;
```

RESULT

StID	StName	SpID	SportID	SportName
2	Bkon	101	101	Chess
3	Ckon	102	102	Carrom
NULL	NULL	NULL	103	Football

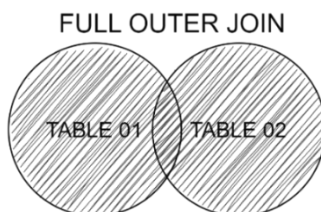
NOTE

If there is no matching record found, the result will be generated as NULL in left hand side.
ගැලපෙන වාර්තාවක් හමු නොවුයේ නම්, වම් පස ප්‍රතිඵලය NULL ලෙස ජනනය වේ.

FULL OUTER JOIN

A FULL OUTER JOIN combines the results of both LEFT JOIN and RIGHT JOIN into a single result set.

FULL OUTER JOIN, LEFT JOIN සහ RIGHT JOIN යන දෙකෙහිම ප්‍රතිඵල තනි ප්‍රතිඵල කාණ්ඩයක් බවට ඒකාබද්ධ කරයි.



It returns all rows when there is a match in either left or right table.

වම්පස හෝ දකුණුපස වගුවේ ගැලපීමක් ඇති විට එය සියලු ජේම් ආපසු ලබා දෙයි.

```
SELECT *
FROM Student
LEFT JOIN Sport
ON Student.SpID=Sport.SportID
UNION
SELECT *
FROM Student
RIGHT JOIN Sport
ON Student.SpID=Sport.SportID;
```

RESULT

StID	StName	SpID	SportID	SportName
1	Akon	100	NULL	NULL
2	Bkon	101	101	Chess
3	Ckon	102	102	Carrom
NULL	NULL	NULL	103	Football

Integrity Constraints

ඒකාබද්ධ සංරෝධක

Rules applied to columns in a table to ensure the accuracy and reliability of the data within the database.

දත්ත සම්ප්‍රදාය තුළ ඇති දත්තවල නිරවද්‍යතාවය සහ විශ්වසනීයත්වය සහතික කිරීම සඳහා වගුවක තීරු සඳහා යොදන රීති වේ.

They are used to enforce data integrity and prevent invalid data entry.

ඒවා දත්ත අඛණ්ඩතාව බලාත්මක කිරීමට සහ වලංගු නොවන දත්ත ඇතුළත් කිරීම් වැළැක්වීමට භාවිතා කරයි.

There are three forms of Integrity constraints.
ඒකාබද්ධ සංරෝධක වල ආකාර තුනක් තිබේ.

1. Entity Integrity

භූතාර්ථ ඒකාබද්ධතාව

Ensures that every row in a table is uniquely identifiable.

වගුවක සෑම පේළියක්ම අනන්‍ය ලෙස හඳුනාගත හැකි බව සහතික කරයි.

Prevents tables from having duplicate rows.
වගු වල අනුපිටපත් වූ පේළි තිබීම වළක්වයි.

Strictly enforces that the identifying column is both UNIQUE (no duplicates) and NOT NULL (cannot be empty).

හඳුනාගැනීමේ තීරුව UNIQUE (අනුපිටපත් නැති) සහ NOT NULL (හිස් විය නොහැක) වන බව දැඩි ලෙස බලාත්මක කරයි.

This identifying column is known as the PRIMARY KEY.

මෙම හඳුනාගැනීමේ තීරුව ප්‍රාථමික යතුර ලෙස හැඳින්වේ.

PRIMARY KEY (Column_name)

2. Referential Integrity

යොමු ඒකාබද්ධතාව

Ensures links between tables are valid.

වගු අතර පවතින සබැඳි වලංගු බව සහතික කරයි.

Validates that every foreign key value entered in the child table corresponds to an existing record in the parent table.

Child වගුවට ඇතුළත් කරනු ලබන සෑම ආගන්තුක යතුරු අගයක්ම parent වගුවේ පවතින පේළියකට අනුරූප වන බව වලංගු කරයි.

If a matching parent record is not found, the referencing is rejected to ensure no invalid links are created between tables.

ගැලපෙන parent වගු පේළියක් සොයාගත නොහැකි වුවහොත්, වගු අතර වලංගු නොවන සබැඳි නිර්මාණය නොවන බව සහතික කිරීම සඳහා යොමු කිරීම ප්‍රතික්ෂේප කරනු ලැබේ.

FOREIGN KEY(Field_name)

REFERENCES Parent_table (Columnn_name)

3. Domain Integrity

වසම් ඒකාබද්ධතාව

Ensures the data inside a specific cell is valid for that column's definition.

නිශ්චිත කෝෂයක් තුළ ඇති දත්ත වීම තීරුවේ අර්ථ දැක්වීමට අනුව වලංගු බව සහතික කරයි.

For example, if a column is meant to hold a person's age, it shouldn't accept text like "Twenty" or impossible numbers like "-5".

උදාහරණයක් ලෙස, තීරුවක් තුළ පුද්ගලයෙකුගේ වයස රඳවා ගැනීමට අදහස් කරන්නේ නම්, ඒ තුළ "විස්ස" වැනි පෙළ හෝ "-5" වැනි තාත්වික නොවන සංඛ්‍යා පැවතිය නොහැක.

Data must strictly follow the defined type, format, and range.

දත්ත විසින් පෙර නිර්වචනය කරන ලද දත්ත වර්ගය, ආකෘතිය සහ පරාසය දැඩි ලෙස අනුගමනය කළ යුතුය.

A list of SQL components that enforce Domain Integrity.

වසම් ඒකාබද්ධතාවය බලාත්මක කරන SQL සංරචක ලැයිස්තුවක්.

1. Data Types දත්ත වර්ග
2. NOT NULL command
3. CHECK command
4. DEFAULT command

SQL Constraints

SQL සම්බාධක

SQL constraints are rules enforced on data columns in a table to ensure data integrity and accuracy. These constraints are defined when a table is created or modified.

SQL සම්බාධක යනු දත්ත අඩුණ්ඩතාව සහ නිරවද්‍යතාවය සහතික කිරීම සඳහා වගුවක ඇති දත්ත තීරු මත බලාත්මක කරන රීති වේ. වගුවක් නිර්මාණය කරන විට හෝ වෙනස් කරන විට මෙම සීමාවන් අර්ථ දක්වා ඇත.

The main types of SQL constraints are listed below.

ප්‍රධාන SQL සම්බාධක වර්ග පහත දැක්වේ.

1. NOT NULL
2. UNIQUE
3. CHECK
4. DEFAULT
5. PRIMARY KEY
6. FOREIGN KEY

NOT NULL

Ensures that a column cannot have a null value. තීරුවකට ශුන්‍ය අගයක් තිබිය නොහැකි බව සහතික කරයි.

1. Applying the NOT NULL constraint when creating a table

වගුවක් සෑදීමේදී NOT NULL සම්බාධකය යෙදීම

SYNTAX

```
CREATE TABLE   (  
        NOT NULL,  
        NOT NULL,  
    .....  
        NOT NULL  
);
```


column name data type

table name

EXAMPLE

Creating the Employee table below. Mentioning all fields is mandatory.

පහත Employee වගුව නිර්මාණය කිරීම. සියලුම ක්ෂේත්‍ර සඳහන් කිරීම අනිවාර්ය වේ.

Employee (RegNo, name, Address, Gender, TelNo)

```
CREATE TABLE Employee (  
    RegNo INT NOT NULL,  
    name VARCHAR(15) NOT NULL,  
    Address VARCHAR(20),  
    Gender VARCHAR(10) NOT NULL,  
    TelNo VARCHAR(10) NOT NULL  
);
```

RESULT

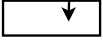
DESCRIBE Employee;

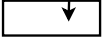
Field	Type	Null	Key	Default	Extra
RegNo	INT	NO		NULL	
name	VARCHAR(15)	NO		NULL	
Address	VARCHAR(20)	YES		NULL	
Gender	VARCHAR(10)	NO		NULL	
TelNo	VARCHAR(10)	NO		NULL	

2. Applying the NOT NULL constraint to an existing table

පවතින වගුවකට NOT NULL සම්බාධකය යෙදීම

SYNTAX

```
ALTER TABLE 
MODIFY COLUMN  
NOT NULL;
```

 table name
 column name  data type

EXAMPLE

It is mandatory to mention the Address field in the Employee table.

Employee වගුවෙහි Address ක්ෂේත්‍රය සඳහන් කිරීම අනිවාර්ය විය යුතුය.

```
ALTER TABLE Employee
MODIFY COLUMN Address Varchar(20) NOT
NULL;
```

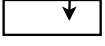
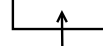
3. Removing the NOT NULL constraint

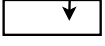
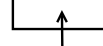
NOT NULL සම්බාධකය ඉවත් කිරීම

Here, the code is written in an equivalent manner without the NOT NULL constraint.

මෙහිදී NOT NULL සම්බාධකය නොදමා සාමාන්‍ය ආකාරයට කේතය ලියනු ලබයි.

SYNTAX

```
ALTER TABLE 
MODIFY COLUMN   ;
```

 table name
 column name  data type

EXAMPLE

Removing the NOT NULL constraint on the Address field in the Employee table.

Employee වගුවේ Address ක්ෂේත්‍රයේ NOT NULL සම්බාධකය ඉවත් කිරීම.

```
ALTER TABLE Employee
MODIFY COLUMN Address Varchar(20);
```

RESULT (Applying NOT NULL)

DESCRIBE Employee;

Field	Type	Null	Key	Default	Extra
RegNo	INT	NO		NULL	
name	VARCHAR(15)	NO		NULL	
Address	VARCHAR(20)	YES		NULL	
Gender	VARCHAR(10)	NO		NULL	
TelNo	VARCHAR(10)	NO		NULL	

RESULT (Removing NOT NULL)

DESCRIBE Employee;

Field	Type	Null	Key	Default	Extra
RegNo	INT	NO		NULL	
name	VARCHAR(15)	NO		NULL	
Address	VARCHAR(20)	NO		NULL	
Gender	VARCHAR(10)	NO		NULL	
TelNo	VARCHAR(10)	NO		NULL	

UNIQUE

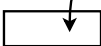
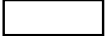
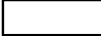
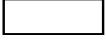
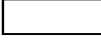
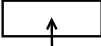
Ensures that all values in a column or a group of columns are unique.

හීරුවක හෝ හීරු සමූහයක ඇති සියලුම අගයන් අනුපිටපත් නොවන බව සහතික කරයි.

1. Applying the UNIQUE constraint when creating a table

වගුවක් සාදීමේදී UNIQUE සම්බාධනය යෙදීම

SYNTAX 01

```
CREATE TABLE  (
    UNIQUE,
    UNIQUE,
  .....
    UNIQUE
);
```

column name data type

EXAMPLE

Creating the Employee table below. All fields except Gender must be unique.

පහත Employee වගුව නිර්මාණය කිරීම. Gender හැර අනෙකුත් සියළුම ක්ෂේත්‍ර අනන්‍ය විය යුතුය.

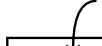
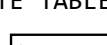
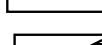
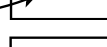
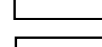
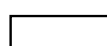
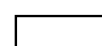
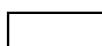
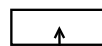
```
CREATE TABLE Employee (
  RegNo INT UNIQUE,
  Name VARCHAR(15) UNIQUE,
  Address VARCHAR(20) UNIQUE,
  Gender VARCHAR(10),
  TelNo VARCHAR(10) UNIQUE
);
```

RESULT

DESCRIBE Employee;

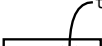
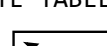
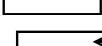
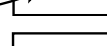
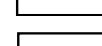
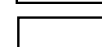
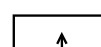
Field	Type	Null	Key	Default	Extra
RegNo	INT	NO		NULL	
name	VARCHAR(15)	NO		NULL	
Address	VARCHAR(20)	NO		NULL	
Gender	VARCHAR(10)	NO		NULL	
TelNo	VARCHAR(10)	NO		NULL	

SYNTAX 02

```
CREATE TABLE  (
    ,
    ,
  .....
    ,
  UNIQUE (  ),
  UNIQUE (  )
);
```

column name data type column name

SYNTAX 03

```
CREATE TABLE  (
    ,
    ,
  .....
  UNIQUE (  ,  ) );
```

column name data type column name

NOTE

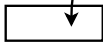
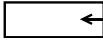
UNIQUE can contain only 1 NULL value.

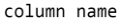
UNIQUE වකෙහි NULL අගයන් 1ක් පමණක් පැවතිය හැක.

2. Applying the UNIQUE constraint to an existing table

පවතින වගුවකට UNIQUE සම්බාධකය යෙදීම

SYNTAX 01

```
ALTER TABLE 
ADD CONSTRAINT 
UNIQUE (  );
```



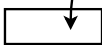
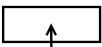
EXAMPLE

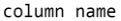
The TelNo field in the Employee table must be unique.

Employee වගුවේ TelNo ක්ෂේත්‍රය අනන්‍ය විය යුතුය.

```
ALTER TABLE Employee
ADD CONSTRAINT uq_TelNo
UNIQUE(TelNo);
```

SYNTAX 02

```
ALTER TABLE 
ADD UNIQUE (  );
```



RESULT (Applying UNIQUE)

DESCRIBE Employee;

Field	Type	Null	Key	Default	Extra
RegNo	INT	YES		NULL	
Name	VARCHAR(15)	YES		NULL	
Address	VARCHAR(20)	YES		NULL	
Gender	VARCHAR(10)	YES		NULL	
TelNo	VARCHAR(10)	YES	UNI	NULL	

RESULT (Removing UNIQUE)

DESCRIBE Employee;

Field	Type	Null	Key	Default	Extra
RegNo	INT	YES		NULL	
Name	VARCHAR(15)	YES		NULL	
Address	VARCHAR(20)	YES		NULL	
Gender	VARCHAR(10)	YES		NULL	
TelNo	VARCHAR(10)	YES		NULL	

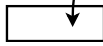
3. Removing the UNIQUE constraint

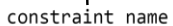
UNIQUE සම්බාධකය ඉවත් කිරීම

To remove a UNIQUE constraint, you need to use the ALTER TABLE statement along with the DROP CONSTRAINT clause.

UNIQUE සම්බාධකය ඉවත් කිරීම සඳහා, ඔබ විසින් ALTER TABLE ප්‍රකාශය සහ DROP CONSTRAINT වගන්තිය භාවිතා කළ යුතුය.

SYNTAX

```
ALTER TABLE 
DROP CONSTRAINT  ;
```



EXAMPLE

Removing the UNIQUE constraint on the TelNo field in the Employee table.

Employee වගුවේ TelNo ක්ෂේත්‍රයේ ඇති UNIQUE සම්බාධකය ඉවත් කිරීම.

```
ALTER TABLE Employee
DROP CONSTRAINT uq_TelNo;
```

DEFAULT

Sets a default value for a column if no value is specified.

අගයක් සඳහන් කර නොමැති නම් තීරුවක් සඳහා පෙරනිම් අගයක් සකසයි.

This helps ensure that a column always has a value, even if the user does not provide one. පරිශීලකයා ලබා නොදුන්නේ, තීරුවකට සෑම විටම අගයක් ඇති බව සහතික කිරීමට මෙය උපකාරී වේ.

1. Applying the DEFAULT constraint when creating a table

වගුවක් සෑදීමේදී DEFAULT සම්බාධකය යෙදීම

SYNTAX

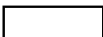
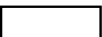
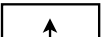
CREATE TABLE  ( ,   DEFAULT ,  );  

table name
column name
data type
default value

EXAMPLE

Creating the Employee table above. Give Nugegoda as the DEFAULT value for the Address field.

ඉහත Employee වගුව නිර්මාණය කිරීම. Address ක්ෂේත්‍රය සඳහා DEFAULT අගය ලෙස Nugegoda ලබා දෙන්න.

Employee (RegNo, Name, Address, Gender, TelNo)

```
CREATE TABLE Employee (  
    RegNo INT,  
    Name VARCHAR(15),  
    Address VARCHAR(20)  
    DEFAULT "Nugegoda",  
    Gender VARCHAR(10),  
    TelNo VARCHAR(10)  
);
```

RESULT

DESCRIBE Employee;

Field	Type	Null	Key	Default	Extra
RegNo	INT	YES		NULL	
Name	VARCHAR(15)	YES		NULL	
Address	VARCHAR(20)	YES		Nugegoda	
Gender	VARCHAR(10)	YES		NULL	
TelNo	VARCHAR(10)	YES		NULL	

2. Applying the DEFAULT constraint to an existing table

පවතින වගුවකට **DEFAULT** සම්බාධකය යෙදීම

SYNTAX

```
ALTER TABLE 
ALTER 
SET DEFAULT  ;
```

table name
COLUMN NAME

default value

EXAMPLE

Set Nugegoda as the DEFAULT value for the Address field in the Employee table.

Employee වගුවේ Address ක්ෂේත්‍රය සඳහා DEFAULT අගය ලෙස Nugegoda ලබා දෙන්න.

```
ALTER TABLE Employee
ALTER Address
SET DEFAULT "Nugegoda";
```

RESULT (Applying DEFAULT)

DESCRIBE Employee;

Field	Type	Null	Key	Default	Extra
RegNo	INT	YES		NULL	
Name	VARCHAR(15)	YES		NULL	
Address	VARCHAR(20)	YES		Nugegoda	
Gender	VARCHAR(10)	YES		NULL	
TelNo	VARCHAR(10)	YES		NULL	

RESULT (Removing DEFAULT)

DESCRIBE Employee;

Field	Type	Null	Key	Default	Extra
RegNo	INT	YES		NULL	
Name	VARCHAR(15)	YES		NULL	
Address	VARCHAR(20)	YES		NULL	
Gender	VARCHAR(10)	YES		NULL	
TelNo	VARCHAR(10)	YES		NULL	

3. Removing the DEFAULT constraint

DEFAULT සම්බාධකය ඉවත් කිරීම

To remove a DEFAULT constraint from a column, use the ALTER TABLE statement along with the DROP DEFAULT clause.

හීරුවකින් DEFAULT සම්බාධකය ඉවත් කිරීමට, DROP DEFAULT වගන්තිය සමඟ ALTER TABLE ප්‍රකාශය භාවිතා කරයි.

SYNTAX

```
ALTER TABLE 
ALTER 
DROP DEFAULT;
```

table name
COLUMN NAME

EXAMPLE

Removing the DEFAULT constraint on the Address field in the Employee table.

Employee වගුවේ Address ක්ෂේත්‍රයේ ඇති DEFAULT සම්බාධකය ඉවත් කිරීම.

```
ALTER TABLE Employee
ALTER Address DROP DEFAULT;
```

CHECK

Ensures that the values in a column satisfy a specific condition.

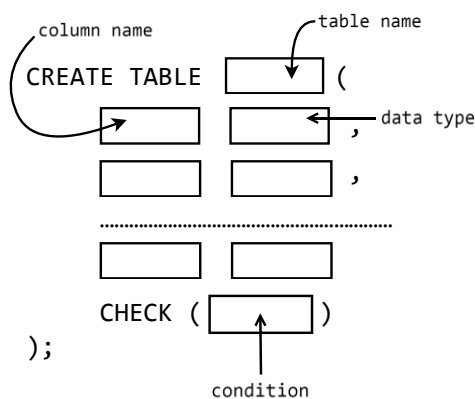
හීරුවක ඇති අගයන් නිශ්චිත කොන්දේසියක් තෘප්තිමත් කරන බව සහතික කරයි.

Helping to enforce domain integrity by restricting the type of data that can be inserted into a table.

වගුවකට ඇතුළු කළ හැකි දත්ත වර්ගය සීමා කිරීමෙන් වසම් අඛණ්ඩතාව බලාත්මක කිරීමට උපකාරී වේ.

1. Applying the CHECK constraint when creating a table

වගුවක් සෑදීමේදී CHECK සම්බාධකය යෙදීම



EXAMPLE

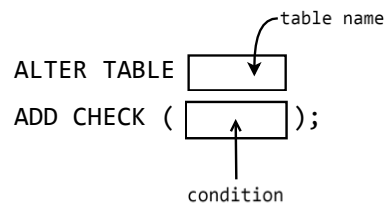
Creating the Employee table above. Age should be checked as 18 years or more.

ඉහත Employee වගුව නිර්මාණය කිරීම. වයස අවුරුදු 18 ට සමානයි හෝ වැඩිද යන්න පරීක්ෂා කළ යුතුය.

```
CREATE TABLE Employee (  
    RegNo INT,  
    Name VARCHAR(15),  
    Address VARCHAR(20),  
    Age INT,  
    Gender VARCHAR(10),  
    TelNo VARCHAR(10),  
    CHECK (Age>=18)  
);
```

2. Applying the CHECK constraint to an existing table

පවතින වගුවකට CHECK සම්බාධකය යෙදීම



EXAMPLE

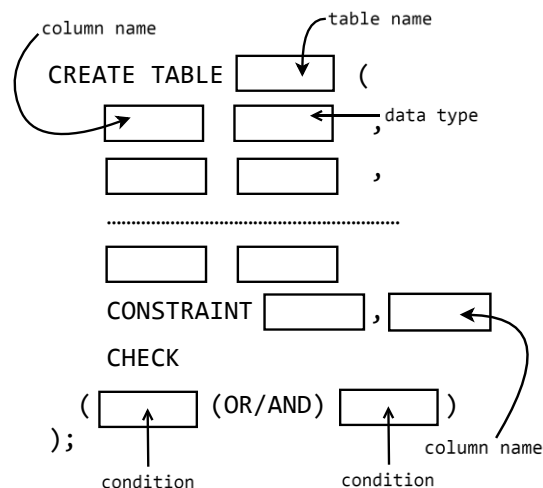
Check whether the age of the members of the Employee table is equal to or greater than 18 years.

Employee වගුවේ සාමාජිකයින්ගේ වයස අවුරුදු 18 ට සමානයි හෝ වැඩිද යන්න පරීක්ෂා කළ යුතුය.

```
ALTER TABLE Employee  
ADD CHECK (Age>=18);
```

3. Creating a CHECK constraint with a name

නමක් සහිතව CHECK සම්බාධකය නිර්මාණය කිරීම



EXAMPLE

Creating the Employee table above. Age above 18 years or address Nugegoda should be checked.

ඉහත Employee වගුව නිර්මාණය කිරීම. වයස අවුරුදු 18 ට වැඩි හෝ ලිපිනය නුගේගොඩද යන්න පරීක්ෂා කළ යුතුය.

Employee (RegNo, Name, Address, Age, Gender, TelNo)

```
CREATE TABLE Employee (  
    RegNo INT,  
    Name VARCHAR(15),  
    Address VARCHAR(20),  
    Age INT,  
    Gender VARCHAR(10),  
    TelNo VARCHAR(10)  
    CONSTRAINT Age, Address CHECK  
    (Age > 18 OR Address = "Nugegoda")  
);
```

4. Removing the CHECK constraint

CHECK සම්බාධකය ඉවත් කිරීම

To remove a named CHECK constraint, you use the ALTER TABLE statement along with the DROP CONSTRAINT clause.

නම් කරන ලද CHECK සම්බාධකය ඉවත් කිරීමට, ඔබ DROP CONSTRAINT වගන්තිය සමඟ ALTER TABLE ප්‍රකාශය භාවිතා කරයි.

```
ALTER TABLE    
DROP CHECK  ;
```

EXAMPLE

Removing the CHECK constraint on the Age field in the Employee table.

Employee වගුවේ Age ක්ෂේත්‍රයේ ඇති CHECK සම්බාධකය ඉවත් කිරීම.

```
ALTER TABLE Employee  
DROP CHECK CHK_EmployeeAge;
```

PRIMARY KEY

A primary key is a field (or combination of fields) in a database table that uniquely identifies each record in that table.

ප්‍රාථමික යතුරක් යනු දත්ත සමූහ වගුවක ඇති ක්ෂේත්‍ර (හෝ ක්ෂේත්‍ර එකතුවක්) එම වගුවේ ඇති සෑම වාර්තාවක්ම අනන්‍ය ලෙස හඳුනා ගනී.

The primary key must contain unique values and cannot contain NULL values.

ප්‍රාථමික යතුරෙහි අනන්‍ය අගයන් අඩංගු විය යුතු අතර NULL අගයන් අඩංගු විය නොහැක.

Its main purpose is to ensure that each record in the table can be uniquely identified and accessed.

එහි ප්‍රධාන අරමුණ වන්නේ වගුවේ ඇති සෑම වාර්තාවක්ම අනන්‍ය ලෙස හඳුනාගෙන ප්‍රවේශ විය හැකි බව සහතික කිරීමයි.

1. Applying the PRIMARY KEY constraint when creating a table

වගුවක් සෑදීමේදී PRIMARY KEY සම්බාධකය යෙදීම

SYNTAX 01

```
CREATE TABLE table name (
    column name data type PRIMARY KEY,
    column name data type ,
    .....
    column name data type
);
```

RESULT

DESCRIBE Employee;

Field	Type	Null	Key	Default	Extra
RegNo	INT	NO	PRI	NULL	
Name	VARCHAR(15)	YES		NULL	
Address	VARCHAR(20)	YES		NULL	
Gender	VARCHAR(10)	YES		NULL	
TelNo	VARCHAR(10)	YES		NULL	

SYNTAX 02

```
CREATE TABLE table name (
    column name data type ,
    column name data type ,
    .....
    column name data type
    PRIMARY KEY ( column name )
);
```

SYNTAX 03

```
CREATE TABLE table name (
    column name data type ,
    column name data type ,
    .....
    column name data type
    CONSTRAINT constraint name
    PRIMARY KEY ( column name )
);
```

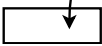
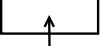
EXAMPLE (SYNTAX 01)

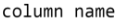
```
CREATE TABLE Employee (
    RegNo INT PRIMARY KEY,
    Name VARCHAR(15),
    Address VARCHAR (20),
    Gender VARCHAR (10),
    TelNo VARCHAR (10)
);
```

2. Applying the PRIMARY KEY constraint to an existing table

පවතින වගුවකට PRIMARY KEY සම්බාධකය යෙදීම

SYNTAX 01

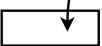
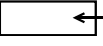
```
ALTER TABLE 
ADD PRIMARY KEY (  );
```

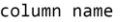


EXAMPLE

```
ALTER TABLE Employee
ADD PRIMARY KEY (RegNo);
```

SYNTAX 02

```
ALTER TABLE 
ADD CONSTRAINT 
PRIMARY KEY (  );
```



RESULT (Applying PRIMARY KEY)

```
DESCRIBE Employee;
```

Field	Type	Null	Key	Default	Extra
RegNo	INT	NO	PRI	NULL	
Name	VARCHAR(15)	YES		NULL	
Address	VARCHAR(20)	YES		NULL	
Gender	VARCHAR(10)	YES		NULL	
TelNo	VARCHAR(10)	YES		NULL	

RESULT (Removing PRIMARY KEY)

```
DESCRIBE Employee;
```

Field	Type	Null	Key	Default	Extra
RegNo	INT	NO		NULL	
Name	VARCHAR(15)	YES		NULL	
Address	VARCHAR(20)	YES		NULL	
Gender	VARCHAR(10)	YES		NULL	
TelNo	VARCHAR(10)	YES		NULL	

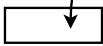
3. Removing the PRIMARY KEY constraint

PRIMARY KEY සම්බාධකය ඉවත් කිරීම

To remove a primary key constraint, use the ALTER TABLE statement along with the DROP PRIMARY KEY clause.

ප්‍රාථමික යතුරු සම්බාධකය ඉවත් කිරීමට DROP PRIMARY KEY වගන්තිය සමඟ ALTER TABLE ප්‍රකාශය භාවිතා කරයි.

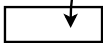
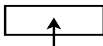
SYNTAX 01

```
ALTER TABLE 
DROP PRIMARY KEY;
```

EXAMPLE

```
ALTER TABLE Employee
DROP PRIMARY KEY;
```

SYNTAX 02

```
ALTER TABLE 
DROP CONSTRAINT  ;
```



Composite Primary Key

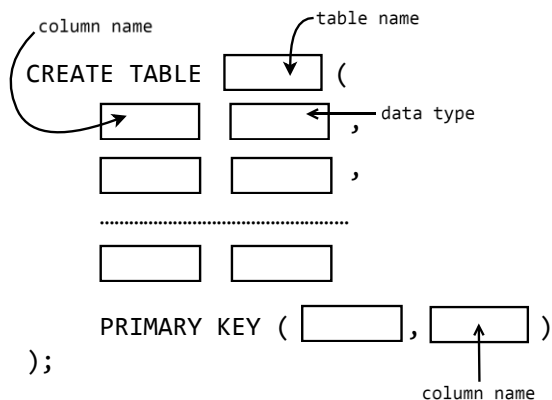
සංයුක්ත ප්‍රාථමික යතුර

A composite primary key consists of two or more columns in a table, used when a single column cannot uniquely identify records.

සංයුක්ත ප්‍රාථමික යතුරක් වගුවක තීරු දෙකකින් හෝ වැඩි ගණනකින් සමන්විත වන අතර, තනි තීරුවකට වාර්තා අනන්‍ය ලෙස හඳුනාගත නොහැකි වීම් භාවිතා වේ.

Creating a composite primary key.

සංයුක්ත ප්‍රාථමික යතුරක් නිර්මාණය කිරීම.



EXAMPLE

Creating the Employee table below. RegNo and Name are Composite Primary Key here.

පහත Employee වගුව නිර්මාණය කිරීම. RegNo හා Name මෙහි සංයුක්ත ප්‍රාථමික යතුර වේ.

Employee (RegNo, Name, Address, Gender, TelNo)

```

CREATE TABLE Employee (
    RegNo INT,
    Name VARCHAR(15),
    Address VARCHAR(20),
    Gender VARCHAR(10),
    TelNo VARCHAR(10),
    PRIMARY KEY(RegNo, Name)
);
  
```

RESULT

DESCRIBE Employee;

Field	Type	Null	Key	Default	Extra
RegNo	INT	NO	PRI	NULL	
Name	VARCHAR(15)	NO	PRI	NULL	
Address	VARCHAR(20)	YES		NULL	
Gender	VARCHAR(10)	YES		NULL	
TelNo	VARCHAR(10)	YES		NULL	

FOREIGN KEY

ආගන්තුක යතුර

A foreign key is a field (or collection of fields) in one table that uniquely identifies a row in another table.

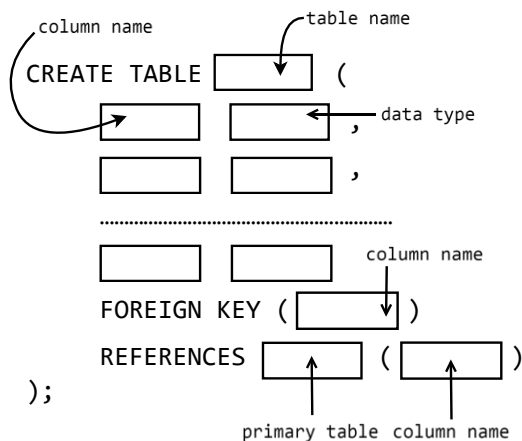
ආගන්තුක යතුරක් යනු එක් වගුවක ඇති ක්ෂේත්‍රයක් (හෝ ක්ෂේත්‍ර එකතුවක්) තවත් වගුවක පේළියක් අනන්‍ය ලෙස හඳුනා ගනී.

It ensures that the value in the foreign key column(s) matches a valid, existing value in the referenced table's primary key.

විදේශීය යතුරු තීරුවෙහි (ය) අගය යොමු කළ වගුවේ ප්‍රාථමික යතුරෙහි වලංගු, පවතින අගයකට ගැළපෙන බව එය සහතික කරයි.

1. Applying the FOREIGN KEY constraint when creating a table

වගුවක් සෑදීමේදී FOREIGN KEY සම්බාධකය යෙදීම



RESULT

DESCRIBE Employee;

Field	Type	Null	Key	Default	Extra
RegNo	INT	NO	PRI	NULL	
Name	VARCHAR(15)	YES		NULL	
Address	VARCHAR(20)	YES		NULL	
Gender	VARCHAR(10)	YES		NULL	
TelNo	VARCHAR(10)	YES		NULL	
DepCode	VARCHAR(5)	YES	MUL	NULL	

EXAMPLE

Creating the Employee table above. The DepCode field is the FOREIGN KEY. It should be related to the DepCode field in the Department table.

ඉහත Employee වගුව නිර්මාණය කිරීම. DepCode ක්ෂේත්‍රය ආගන්තුක යතුර වේ. එය Department වගුවේ DepCode ක්ෂේත්‍රය හා සම්බන්ධ විය යුතුය.

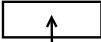
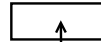
```

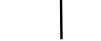
CREATE TABLE Employee (
    RegNo INT,
    Name VARCHAR(15),
    Address VARCHAR(20),
    Gender VARCHAR(10),
    TelNo VARCHAR(10),
    DepCode VARCHAR(5),
    FOREIGN KEY(DepCode) REFERENCES
    Department(DepCode)
);

```

Applying the FOREIGN KEY constraint to an existing table

පවතින වගුවකට FOREIGN KEY සම්බන්ධතා යෙදීම

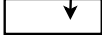
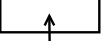
```
ALTER TABLE 
ADD FOREIGN KEY (  )
REFERENCES  (  )
);
```

 primary table  column name

```
ALTER TABLE Employee
ADD FOREIGN KEY (DepCode)
REFERENCES Department(DepCode));
```

Removing the FOREIGN KEY constraint

FOREIGN KEY සම්බන්ධතා ඉවත් කිරීම

```
ALTER TABLE 
DROP FOREIGN KEY  ;
```

 column name

EXAMPLE

Remove the existing FOREIGN KEY of the Employee table, DepCode.

Employee වගුවේ දැනට ඇති ආගන්තුක යතුර වන DepCode ඉවත් කරන්න.

```
ALTER TABLE Employee
DROP FOREIGN KEY DepCode;
```

RESULT (Applying FOREIGN KEY)

DESCRIBE Employee;

Field	Type	Null	Key	Default	Extra
RegNo	INT	NO	PRI	NULL	
Name	VARCHAR(15)	YES		NULL	
Address	VARCHAR(20)	YES		NULL	
Gender	VARCHAR(10)	YES		NULL	
TelNo	VARCHAR(10)	YES		NULL	
DepCode	VARCHAR(5)	YES	MUL	NULL	

RESULT (Removing FOREIGN KEY)

DESCRIBE Employee;

Field	Type	Null	Key	Default	Extra
RegNo	INT	NO	PRI	NULL	
Name	VARCHAR(15)	YES		NULL	
Address	VARCHAR(20)	YES		NULL	
Gender	VARCHAR(10)	YES		NULL	
TelNo	VARCHAR(10)	YES		NULL	
DepCode	VARCHAR(5)	YES	MUL	NULL	