Some basic SQL queries

1. Create Database

   CREATE DATABASE databaseName;

   Eg. CREATE DATABASE testDb;

2. Drop Database

   DROP DATABASE databaseName;

   Eg. DROP DATABASE testDb;

3. Create Table

i. Simple Table

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

ii. Table with SQL Constrain

The following constraints are commonly used in SQL:

- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Uniquely identifies a row/record in another table
- CHECK - Ensures that all values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column when no value is specified
- INDEX - Used to create and retrieve data from the database very quickly

```
CREATE TABLE table_name (
   column1 datatype constraint,
   column2 datatype constraint,
   column3 datatype constraint,
   ....
);
```

Aakrit Subedi
contactme@aakritsubedi.com.np
aakritsubedicom.np

Eg.

a.
```
CREATE TABLE student{
Id int,
First_name varchar(25),
Last_name varchar(25)
}
```

b.
```
CREATE TABLE Persons (
  ID int NOT NULL,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255) NOT NULL,
  Age int
);
```

c.
```
CREATE TABLE Persons (
  ID int NOT NULL,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Age int,
  UNIQUE (ID)
);
```

d.
```
CREATE TABLE Persons (
  ID int NOT NULL,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Age int,
  PRIMARY KEY (ID)
);
```

e.
```
CREATE TABLE Orders (
  OrderID int NOT NULL,
  OrderNumber int NOT NULL,
  PersonID int,
  PRIMARY KEY (OrderID),
  FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

f.
```
CREATE TABLE Persons (
  ID int NOT NULL,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
 City varchar(255) DEFAULT 'Sandnes’,
  Age int,
  CHECK (Age>=18)
);
```

4. Drop table

   DROP TABLE *table_name*;

   Eg. DROP TABLE student;

5. Alter table
   i. Add Column

      ALTER TABLE *table*_name
      ADD column_name datatype;

      ALTER TABLE student
      ADD father_name varchar(50);

   ii. Drop Column

      ALTER TABLE *table_name*
      DROP COLUMN *column_name*;

      ALTER TABLE student
      DROP COLUMN mother_name;

   iii. Alter/Modify Column

      ALTER TABLE *table_name*
      MODIFY COLUMN *column_name datatype*;

6. SQL Index
   Indexes are used to retrieve data from the database very fast. The users cannot see the indexes; they are just used to speed up searches/queries.

   CREATE INDEX *index_name*
   ON *table_name* (*column1*, *column2*, ...);

   Eg. The SQL statement below creates an index named "idx_lastname" on the "LastName" column in the "Persons" table:

   ```
   CREATE INDEX idx_lastname
   ON Persons (LastName);
   ```

If you want to create an index on a combination of columns, you can list the column names within the parentheses, separated by commas:

```
CREATE INDEX idx_pname
ON Persons (LastName, FirstName);
```

<u>DROP INDEX Statement</u>

ALTER TABLE *table_name*
DROP INDEX *index_name*;

Aakrit Subedi
contactme@aakritsubedi.com.np
aakritsubedicom.np

7. Auto Increment

```
CREATE TABLE Persons (
  ID int NOT NULL AUTO_INCREMENT,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Age int,
  PRIMARY KEY (ID)
);
```

8. SQ L date data type

MySQL comes with the following data types for storing a date or a date/time value in the database:

- DATE - format YYYY-MM-DD
- DATETIME - format: YYYY-MM-DD HH:MI:SS
- TIMESTAMP - format: YYYY-MM-DD HH:MI:SS
- YEAR - format YYYY or YY

9. VIEWS

In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Eg.

```
CREATE VIEW [Current Product List] AS
SELECT ProductID, ProductName
FROM Products
WHERE Discontinued = No;
```

10. SELECT

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

Aakrit Subedi
contactme@aakritsubedi.com.np
aakritsubedicom.np

Eg. SELECT CustomerName,City FROM Customers;

11. SELECT DISTINCT
    SELECT DISTINCT *column1*, *column2, ...* FROM *table_name*;

    Eg. SELECT DISTINCT Country FROM Customers;

12. WHERE
    SELECT *column1*, *column2, ...* FROM *table_name* WHERE *condition*;

    *Eg. SELECT * FROM Customers WHERE Country='Mexico';*

13. AND, OR and NOT Operators
    SELECT *column1*, *column2, ...* FROM *table_name*
    WHERE *condition1* AND *condition2* OR *condition3* *NOT condition3*;

    Eg. SELECT * FROM Customers
    WHERE Country='Germany' OR Country='Nepal' AND NOT Country='USA';

14. ORDER BY
    SELECT *column1*, *column2, ...* FROM *table_name* ORDER BY *column1, column2, ...* ASC|DESC;
    Eg. SELECT * FROM student ODER BY first_name DESC;

15. INSERT INTO
    INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
    VALUES (*value1*, *value2*, *value3*, ...);

    Eg. INSERT INTO Customers (CustomerName, ContactName, Address, City, Country)
    VALUES ('Aakrit', Subedi', 'Basundhara', 'Kathmandu', '', 'Nepal');

16. NOT NULL
    A field with a NULL value is a field with no value.

    SELECT *column_names* FROM *table_name* WHERE *column_name* IS NULL;

    Eg. SELECT LastName, FirstName, Address FROM Persons
    WHERE Address IS NULL;

    SELECT *column_names* FROM *table_name* WHERE *column_name* IS NOT NULL;

    Eg. SELECT LastName, FirstName, Address FROM Persons
    WHERE Address IS NOT NULL;

17. UPDATE

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

Eg. UPDATE Customers SET ContactName = 'Aakrit Subedi', City= 'Kathmandu'
WHERE CustomerID = 1;

18. DELETE
DELETE FROM *table_name* WHERE *condition*;

Eg. DELETE FROM Customers WHERE CustomerName='Aakrit Subedi';

19. LIMIT
SELECT *column_name(s)* FROM *table_name* WHERE *condition* LIMIT *number*;

Eg. SELECT * FROM student WHERE result='Pass' LIMIT 5;

20. MIN() and MAX()
SELECT MIN(*column_name*) FROM *table_name* WHERE *condition*;

Eg. SELECT MIN(salary) FROM student where department='computer';

SELECT MAX(*column_name*) FROM *table_name* WHERE *condition*;

Eg. SELECT MAX(salary) FROM student where department='computer';

21. COUNT() , AVG() and SUM()
SELECT COUNT(*column_name*)FROM *table_name* WHERE *condition*;

SELECT AVG(*column_name*) FROM *table_name* WHERE *condition*;

SELECT SUM(*column_name*) FROM *table_name* WHERE *condition*;

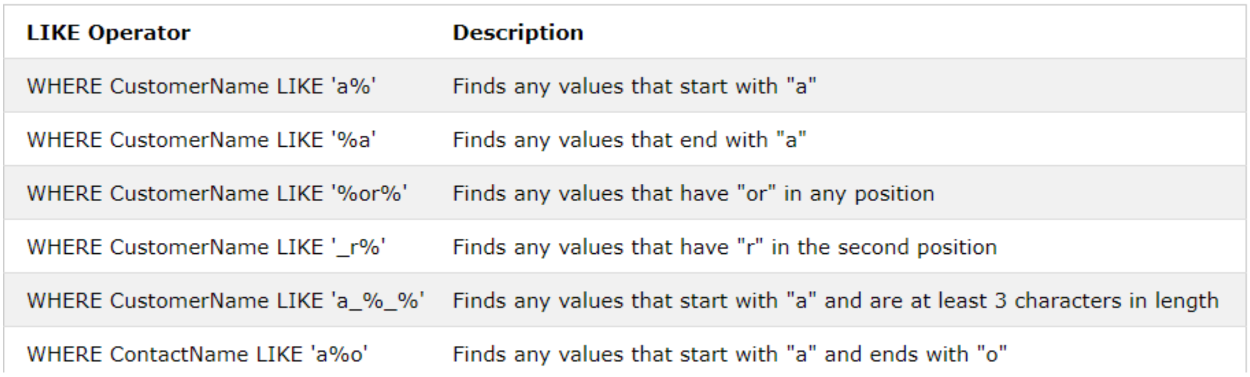
22. Like

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.There are two wildcards used in conjunction with the LIKE operator:

- % - The percent sign represents zero, one, or multiple characters
- _ - The underscore represents a single character

SELECT *column1, column2, ...* FROM *table_name* WHERE *column* LIKE *pattern*;

Aakrit Subedi
contactme@aakritsubedi.com.np
aakritsubedicom.np



LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%_%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

23. IN Operator
    The IN operator allows you to specify multiple values in a WHERE clause.
    SELECT *column_name(s)* FROM *table_name* WHERE *column_name* IN (*value1*, *value2*, ...);
    Eg.
    *SELECT * FROM Customers WHERE Country IN ('Nepal', 'India', 'UK');*

24. Between Operator
    The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

    SELECT column_name(s) FROM table_name
    WHERE column_name BETWEEN value1 AND value2;

    *Eg. SELECT * FROM Products WHERE Price BETWEEN 10 AND 20;*

25. Aliases
    SQL aliases are used to give a table, or a column in a table, a temporary name. Aliases are often used to make column names more readable. An alias only exists for the duration of the query.

    a. Alias Column Syntax

       SELECT *column_name* AS *alias_name* FROM *table_name;*
       Eg.
       SELECT id AS roll_no FROM student;

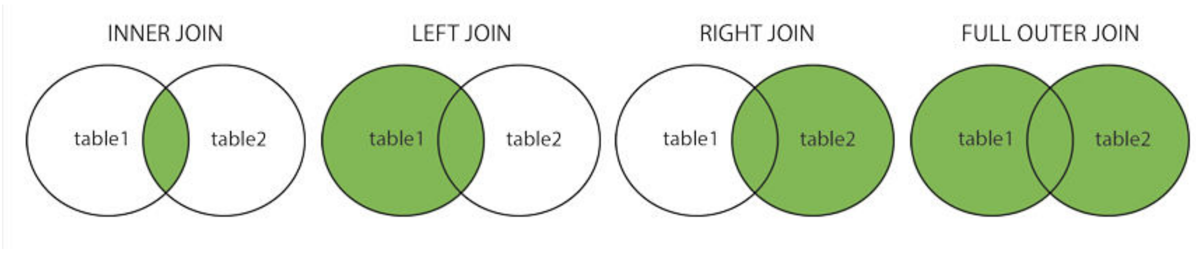
    b. Alias Table Syntax
       SELECT *column_name(s)* FROM *table_name* AS *alias_name;*

       Eg. We use the "Customers" and "Orders" tables, and give them the table aliases of "c" and "o" respectively.
       SELECT o.OrderID, o.OrderDate, c.CustomerName
       FROM Customers AS c, Orders AS o
       WHERE c.CustomerName="Around the Horn" ANDc.CustomerID=o.CustomerID;

Aliases can be useful when:

- There are more than one table involved in a query
- Functions are used in the query
- Column names are big or not very readable
- Two or more columns are combined together

26. Joins
    A JOIN clause is used to combine rows from two or more tables, based on a related column between them.
    Different Types of SQL JOINs
    Here are the different types of the JOINs in SQL:
    - (INNER) JOIN: Returns records that have matching values in both tables
    - LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table
    - RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table
    - FULL (OUTER) JOIN: Return all records when there is a match in either left or right table



a. Inner Join
   The INNER JOIN keyword selects records that have matching values in both tables.

   SELECT *column_name(s)* FROM *table1*
   INNER JOIN *table2* ON *table1.column_name* = *table2.column_name*;

   Eg. SELECT Orders.OrderID, Customers.CustomerName FROM Orders
   INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

   Join Three Table

   SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName
   FROM ((Orders
   INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID)
   INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);

b. Left Join
   The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.

   SELECT *column_name(s)* FROM *table1*
   LEFT JOIN *table2* ON *table1.column_name = table2.column_name*;

   Eg. SELECT Customers.CustomerName, Orders.OrderID FROM Customers
   LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
   ORDER BY Customers.CustomerName;

c. Right Join
   The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

   SELECT *column_name(s)* FROM *table1*
   RIGHT JOIN *table2* ON *table1.column_name = table2.column_name*;

   Eg. SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
   FROM Orders
   RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
   ORDER BY Orders.OrderID;

d. Full Outer Join
   The FULL OUTER JOIN keyword return all records when there is a match in either left (table1) or right (table2) table records.

   SELECT *column_name(s)* FROM *table1*
   FULL OUTER JOIN *table2* ON *table1.column_name = table2.column_name*;

   Eg. SELECT Customers.CustomerName, Orders.OrderID
   FROM Customers
   FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
   ORDER BY Customers.CustomerName;

e. Self-Join
   A self JOIN is a regular join, but the table is joined with itself.

   SELECT *column_name(s)*
   FROM *table1 T1, table1 T2*
   WHERE *condition*;

   SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City
   FROM Customers A, Customers B
   WHERE A.CustomerID <> B.CustomerID

Aakrit Subedi
contactme@aakritsubedi.com.np
aakritsubedicom.np

```
AND A.City = B.City
ORDER BY A.City;
```

27. Union Operator

The UNION operator is used to combine the result-set of two or more SELECT statements.

- Each SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in each SELECT statement must also be in the same order

SELECT *column_name(s)* FROM *table1*
UNION ALL
SELECT *column_name(s)* FROM *table2*;

Eg.

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
ORDER BY City;
```

```
SELECT City FROM Customers
UNION ALL
SELECT City FROM Suppliers
ORDER BY City;
```

```
SELECT City, Country FROM Customers
WHERE Country='Germany'
UNION
SELECT City, Country FROM Suppliers
WHERE Country='Germany'
ORDER BY City;
```

```
SELECT City, Country FROM Customers
WHERE Country='Germany'
UNION ALL
SELECT City, Country FROM Suppliers
WHERE Country='Germany'
ORDER BY City;
```

28. Group By
    The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
GROUP BY *column_name(s)*
ORDER BY *column_name(s);*

Eg.
```
SELECT COUNT(CustomerID), Country
    FROM Customers
    GROUP BY Country;
```

Aakrit Subedi
contactme@aakritsubedi.com.np
aakritsubedicom.np

29. Having Clause
    The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
GROUP BY *column_name(s)*
HAVING *condition*
ORDER BY *column_name(s);*

Eg. SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;

30. Exists Operator
    The EXISTS operator is used to test for the existence of any record in a subquery. The EXISTS operator returns true if the subquery returns one or more records.

SELECT *column_name(s)*
FROM *table_name*
WHERE EXISTS
(SELECT *column_name* FROM *table_name* WHERE *condition*);

Aakrit Subedi
contactme@aakritsubedi.com.np
aakritsubedicom.np

Aakrit Subedi
contactme@aakritsubedi.com.np
aakritsubedicom.np

Aakrit Subedi
contactme@aakritsubedi.com.np
aakritsubedicom.np

Aakrit Subedi
contactme@aakritsubedi.com.np
aakritsubedicom.np