What is a database?
Database is a collection of data and stores this data in the form of a table.

What is a table?
Table is a data structure which stores the data in the form of rows and columns.

What is RDBMS?
RDBMS stands for Relational Database Management System.The data in RDBMS is stored in database objects called tables.

What is the field?
Fields are nothing but columns of the table. Every table is broken up into smaller entities called fields.

What is SQL?
Structured query language is used for handling(retrieving ,modifying,deleting) structured data. Using SQL you can easily create and manipulate the database, access and modify the table rows and columns.

Relational Vs Non-Relational database:

Relational Database:
In relational database data is stored in tables and also the tables have relation between them.
In relational databases the schema is predefined, fixed, and static.
EX-MY-SQL,Oracle

Non-Relational Database:
In a non-relational database data is stored in the form Key-value or Document or graph.
A non-relational database schema is dynamic for unstructured data.
EX-Mongodb,No-SQL

SQL Commands:

Data Definition Language(DDL):
1] CREATE:- used to create a database and table.
2] ALTER:- used to modify existing databases and tables.
3] DROP:-used to delete the entire table.
4]**Truncate:**- Used to delete all records from the table but doesn’t delete the table schema.
5]Rename:- used to rename the column names

Data Manipulation Language(DML):

1]SELECT:-used to select or retrieve the records from table
2]INSERT:-Create a record
3]UPDATE:-update a record
4]DELETE:-deletes the record

Data Control Language(DCL):

1] GRANT:-used to give privilege to the user.
2] REVOKE:-used to take back privileges granted from users.

Transaction control Language(TCL):

Transaction control language commands are used to modify the transactions of the database.
1] Commit: This command is used to permanently save any transaction into the database.
2] Rollback: This command restores the database to last commited state
3] Savepoint: command is used to temporarily save a transaction so that you can rollback to that point whenever required.

SQL Constraint:

Constraints are the rules enforced on data columns of the table. These are used to limit the type of data that can go into a table.
Constraints can either be column level or table level. Column level constraints are applied only to one column whereas, table level constraints are applied to the entire table.

Following are some of the most commonly used constraints available in SQL:

1] NOT NULL Constraint: Ensures that a column cannot have a NULL value.
2] DEFAULT Constraint: Provides a default value for a column when none is specified.
3] UNIQUE Constraint: Ensures that all the values in a column are different.
4] PRIMARY Key: Uniquely identifies each row/record in a database table.
5] FOREIGN Key: Uniquely identifies a row/record in any other database table.
6] CHECK Constraint: The CHECK constraint ensures that all values in a column satisfy certain conditions.
7] INDEX: Used to create and retrieve data from the database very quickly

Primary Key and Unique key :

You can have only one primary key and the primary key cannot hold null values.
We should use a primary key when we have to uniquely identify the record
Unique key can hold NULL value. The purpose of a unique key is to make sure the values do not duplicate.
Primary can be a single column or combined two columns which is called a composite key.

(Student_id),(student_id,project_id)

Foreign Key:

A foreign key constraint is used to prevent actions that would destroy links between two tables.
a foreign key is a field or a column that is used to establish a link between two tables.
A foreign key is a field in one table that refers to the primary key in another table.
The table with the primary key is called the parent or referenced table and the table with foreign key is the child table.

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

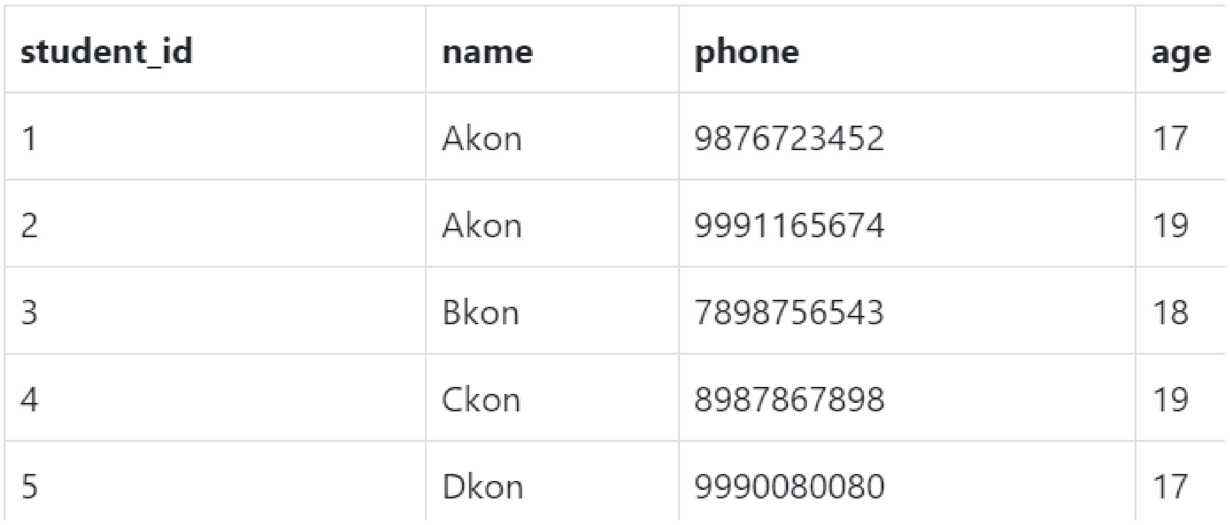
Database Keys

Keys in the databases are used to identify the records uniquely and they also help us to establish the relationship between tables.

A Key can be a single attribute or a group of attributes, where the combination may act as a key.

Why do we need a key?

In real-world applications data is huge and when there are duplicates present in the data it will be very difficult to retrieve the data so to identify records uniquely we need keys.



student_id	name	phone	age
1	Akon	9876723452	17
2	Akon	9991165674	19
3	Bkon	7898756543	18
4	Ckon	8987867898	19
5	Dkon	9990080080	17

Super Key:

Super Key is defined as a set of attributes r columns within a table that can uniquely identify each record within a table. Super Key is a superset of Candidate key.

Super keys is nothing but the combination of all the keys using which we can identify each record uniquely.

In the table defined above super key would include student_id, (student_id, name), phone etc.

Confused? The first one is pretty simple as student_id is unique for every row of data, hence it can be used to identity each row uniquely.

Next comes, (student_id, name), now name of two students can be same, but their student_id can't be same hence this combination can also be a key.

Similarly, phone number for every student will be unique, hence again, phone can also be a key.

So they all are super keys.

Candidate Key:

Candidate keys are defined as the minimal set of fields that can uniquely identify each record in a table.
It is an attribute or a set of attributes that can act as a Primary Key for a table to uniquely identify each record in that table.

There can be more than one candidate key.
In our example, student_id and phone both are candidate keys for table Student.

A candidate key can never be NULL or empty. And its value should be unique.

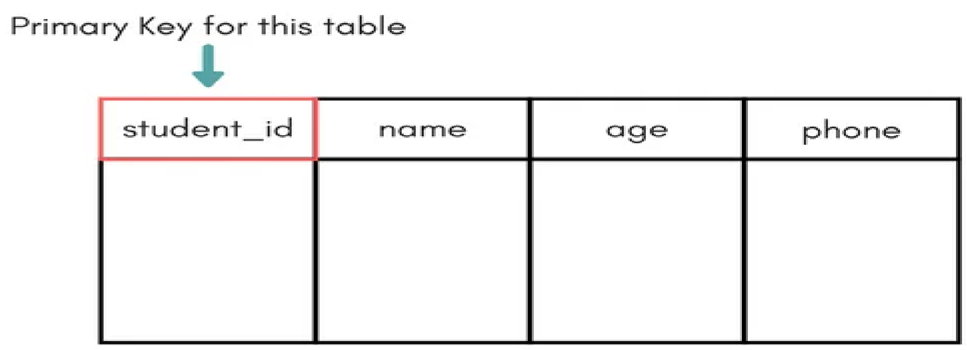
There can be more than one candidate key for a table.

A candidate key can be a combination of more than one column (attributes).

Primary Key:

A primary key is a candidate key that is most appropriate to become the main key for any table. It is a key that can uniquely identify each record in a table.

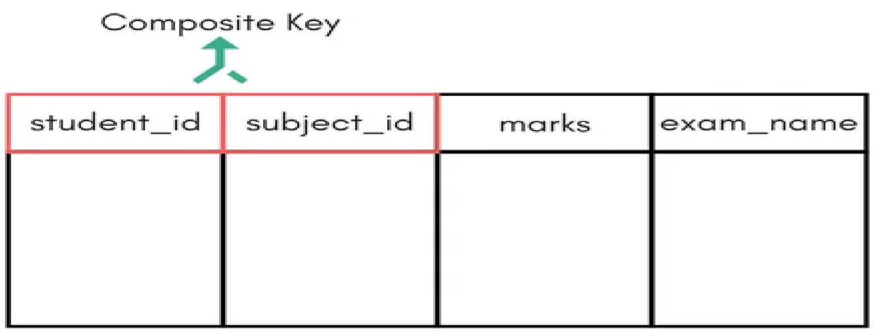
```
CREATE TABLE Students (    /* Create table with a single field as primary key */
   ID INT NOT NULL
   Name VARCHAR(255)
   PRIMARY KEY (ID)
);
```



Composite Key:

Key that consists of two or more attributes that uniquely identify any record in a table is called a Composite key. But the attributes which together form the Composite key are not a key independently or individually.

A composite key is nothing but when we have two columns combined as the primary key Ex- (order_id,order_date)



Score Table – To save scores of the student for various subjects.

In the above picture we have a Score table which stores the marks scored by a student in a particular subject.

In this table student_id and subject_id together will form the primary key, hence it is a composite key.

Secondary or Alternative key:
The candidate key which are not selected as primary key are known as secondary keys or alternative keys.

Non-key Attributes:
Non-key attributes are the attributes or fields of a table, other than candidate key attributes/fields in a table.

Non-prime Attributes:
Non-prime Attributes are attributes other than Primary Key attribute(s)..

UNIQUE constraint?
A UNIQUE constraint ensures that all values in a column are different. This provides uniqueness for the column(s) and helps identify each row uniquely. Unlike primary key, there can be multiple unique constraints defined per table. The code syntax for UNIQUE is quite similar to that of PRIMARY KEY and can be used interchangeably.

Types of Joins:
The SQL Join clause is used to combine records (rows) from two or more tables in a SQL database based on a related column between the two.

There are four different types of JOINs in SQL:

(INNER) JOIN: Retrieves records that have matching values in both tables involved in the join. This is the widely used join for queries.

LEFT (OUTER) JOIN: Retrieves all the records/rows from the left and the matched records/rows from the right table.

RIGHT (OUTER) JOIN: Retrieves all the records/rows from the right and the matched records/rows from the left table.

FULL (OUTER) JOIN: Retrieves all the records where there is a match in either the left or right table.

Data Integrity:

Data Integrity is **used to maintain the Accuracy and consistency of data in the Table**.

Self-Join:
A self JOIN is a case of regular join where a table is joined to itself based on some relation between its own column(s). Self-join uses the INNER JOIN or LEFT JOIN clause and a table alias is used to assign different names to the table within the query.

Cross-Join:

Cross join can be defined as a cartesian product of the two tables included in the join. The table after join contains the same number of rows as in the cross-product of the number of rows in the two tables. If a WHERE clause is used in cross join then the query will work like an INNER JOIN.

Normalization:

Database Normalization is a technique of organizing the data in the database.

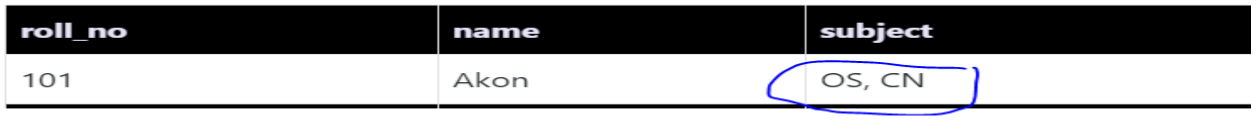
Normalization is used for mainly two purposes,

1] Eliminating redundant(useless) data.
2] Ensuring data dependencies make sense i.e data is logically stored.

First Normal Form:

If tables in a database are not even in the 1st Normal Form, it is considered as bad database design.

The first normal form should follow the below rules:
1] All columns should have a single attribute value i.e single value



roll_no	name	subject
101	Akon	OS, CN

2] The value stored in a column should be of the same type.
3] each column should have a unique name.

Second Normal Form:

For a table to be in second normal form it should follow the below condition:
1] A table should be in first normal form.
2[There should be no partial dependency.

Partial Dependency exists, when for a composite primary key, any attribute in the table depends only on a part of the primary key and not on the complete primary key.

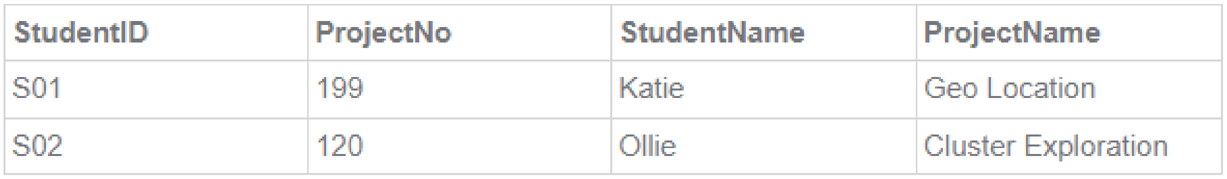
To remove Partial dependency, we can divide the table, remove the attribute which is causing partial dependency, and move it to some other table where it fits in well.

What is a dependency?

Dependency means when all the columns of the table are dependent on a single column which we can say primary key.
Ex- student_id - primary key
Now using student_id we can fetch the student name, email, and phone so here all columns are dependent on student_id.

Partial Dependency:(https://www.studytonight.com/dbms/second-normal-form.php)

Partial dependency means when Non_Prime attributes are dependent on part of the candidate key.



StudentID	ProjectNo	StudentName	ProjectName
S01	199	Katie	Geo Location
S02	120	Ollie	Cluster Exploration

In the above table we have (StudentID, ProjectNo) as a candidate key.
Using candidate key we can uniquely identify the record.
Here projectName is dependent on ProjectNO and it has nothing to do with StudentID so it is only depdendent on part of the candidate key.

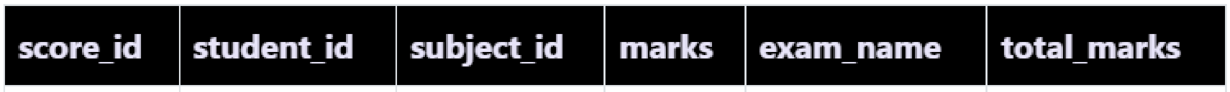
To make above table in second normal form. We should have another table with ProjectNo and ProjectName. And in the first table, we should have StudentId, StudendentName, and ProjectNo.

Third Normal Form:

For a table to be in the third normal form,

1. It should be in the Second Normal form.
2. And it should not have Transitive Dependency.

This is Transitive Dependency. When a non-prime attribute or non-key depends on other non-prime attributes rather than depending upon the prime attributes or primary key.



score_id	student_id	subject_id	marks	exam_name	total_marks

The primary key for our Score table is a composite key → student_id + subject_id.

Our new column exam_name depends on both student and subject.
So we can say that exam_name is dependent on both student_id and subject_id.

And what about our second new column total_marks? Does it depend on our Score table's primary key?

Well, the column total_marks depends on exam_name as with exam type the total score changes. For example, practicals are fewer marks while theory exams are of more marks.

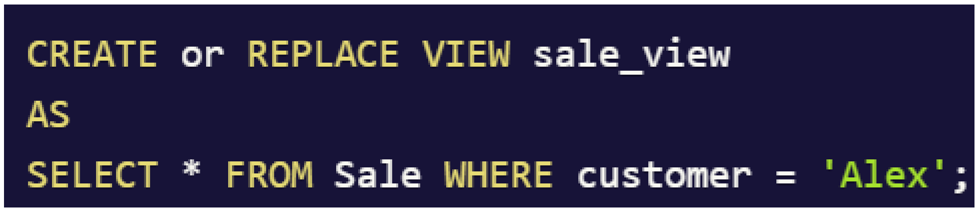
But, exam_name is just another column in the score table. It is not a primary key or even a part of the primary key, and total_marks depend on it.

The solution to remove the transitive dependency.
Again the solution is very simple. Take out the columns exam_name and total_marks from Score table and put them in an Exam table and use the exam_id wherever required.

Boyce -Codd Normal Form:(BCNF)

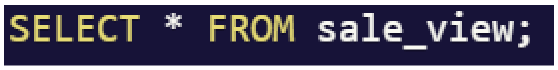
Views:

A view in SQL is a logical subset of data from one or more tables. View is used to restrict data access.



```
CREATE or REPLACE VIEW sale_view
AS
SELECT * FROM Sale WHERE customer = 'Alex';
```

Display the view data:



```
SELECT * FROM sale_view;
```

Types of view:

1] Simple view: This is created from one table and doesnt contain group data.
2] Complex view: This is created from multiple table and contains group data.

Stored Procedures:

A stored procedure is a block of sql code that we save and used it over and over again to avoid repetitive tasks.

So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

Stored procedure are made up of following section:

1] Declarative section: In this section, variables, constants, curser or exceptions that are going to be used by procedure or function are declared.

2] Executable section: In this section, the definition of procedure or function created is written. This section also contains the SQL or PL/SQL statements assigning values, controlling execution and manipulating data**.**

3] Exception Handling: In this section, the expected exceptions are written which may arise during execution of code written in executable part. This section is optional.

Creating Stored Procedure:

```
CREATE PROCEDURE SelectAllCustomers
AS
SELECT * FROM Customers
GO;
```

Executing stored procedure:

```
EXEC SelectAllCustomers;
```

Stored Procedure Vs Functions:

Stored procedure may or may not return the values.
Functions always return value when they called.

Stored procedure can not called from Function block
Functions can be called from stored procedures.

Trigger:

Trigger is code or program which automatically execute with response to some event on the table.

Mainly trigger helps to maintain the integrity of the database.
Ex- when a new student is added to the student database new records should be created in related tables such as Exam,Score and Attendance tables.

Indexes:

The index is a performance tunning method for allowing faster retrieval of records from the table. An index creates an entry for each value and makes it faster to retrieve data.

We have three types of indexes:
1] Unique index: This indexing does allow columns to have duplicates.
2] Clustered index: This type of index reorders the physical order of the table and searches based on the key values.
3] Non-Clustered index: This type of index does not alter the physical order of the table and maintains the logical order of data. each table can have 999 non-clustered indexes.

Online Analytical Processing Vs Online transaction processing:

OLAP is a software tool that provides analysis of data for business decisions.OLAP system allows users to analyze the database information from multiple database system at one time The primary objective is data analysis and not data processing.

OLTP supports transaction-oriented applications in a 3-tier architecture.OLTP administers the day-to-day transactions of an organization.
The primary objective is a data processing and not data analysis. Unlike the OLAP system the goal of the OLTP system is to serve real-time transactions.

ACID(Atomicity, Consistency, Isolation, Durability):

Acid properties are used to for maintaining the integrity of the database during transaction processing.
Atomicity: A transaction is a single unit of operation you either execute it entirely or do not execute it at all. There cannot be partial execution.
Consistency: Once the transaction is executed it should move from one consistent state to another.

Isolation: The transaction should be executed in isolation from other transaction. During concurrent transactions, intermediate transaction results from simultaneously executed transactions should not be made available to each other.
Durability: After the successful completion of a transaction the changes in the database should persist even in the case of system failures.

SQL Database Commands:

Creating and deleting the database:

1] **SHOW DATABASES;**
2] **CREATE DATABASE** Database_Name;
3] **DROP DATABASE** Database_Name;

SQL Table Commands:

1] **SHOW TABLES;**

2] **CREATE TABLE** table_Name;

EX-CREATE TABLE Employee
(EmployeeID int,FirstName varchar(255));

3] **DESCRIBE** table_name;

EX-DESC Employee;

Starter example:

```
EX-CREATE DATABASE company;
     Show databases;
     USE company;

CREATE TABLE employee(
     empid int NOT NULL AUTO_INCREMENT,name varchar(20) NOT NULL,age int NOT NULL,loc NOT NULL
DEFAULT "Goa",Email_id varchar(20),PRIMARY KEY(empid),UNIQUE KEY(email_id));

INSERT INTO employee(name,age,loc,email)
VALUES('om',22,'pune,"ab@gmail.com"),('kali',23,'Hyd','xy@gmail.com');

SELECT * FROM employee;
```

4] **DROP TABLE** table_Name;

Ex-DROP TABLE Employee;

5] **DELETE FROM** table_name [**WHERE** condition];

The DELETE statement is used to delete rows from a table. If you want to remove a specific row from a table you should use WHERE condition.

EX-DELETE FROM Employee WHERE emp_id=2;

EX-DELETE FROM Employee;

If we don't specify where condition it will delete all records from table;

6] TRUNCATE TABLE table_name;

Drop table command can also be used to delete the complete table but it deletes the table structure too. TRUNCATE TABLE doesn't delete the structure of the table.

Ex-TRUNCATE TABLE Employee;

7] COPY TABLE

Ex-SELECT * INTO New_table_name FROM old_table_name;

8] ALTER TABLE

The ALTER TABLE statement in SQL allows you to add, modify, and delete columns of an existing table.This statement also allows database users to add and remove various SQL constraints on the existing tables.

a] ALTER TABLE Add Column statement

EX- ALTER TABLE table_name ADD column_name column-definition;

EX- ALTER TABLE Employee ADD salary Varchar(20);

b] ALTER TABLE change data type of Column statement

EX- ALTER TABLE table_name ALTER COLUMN column_name column-definition;

EX- ALTER TABLE Employee ALTER COLUMN salary Int;

c] ALTER TABLE Drop Column statement

EX- ALTER TABLE table_name DROP COLUMN column_name;

EX- ALTER TABLE Employee DROP COLUMN salary ;

d] ALTER TABLE Rename Column statement

EX- ALTER TABLE table_name RENAME COLUMN old_column_name to new_column_name;

EX- ALTER TABLE Employee RENAME COLUMN salary to emp_salary;

String Functions:

1] length()

This function returns the length of the string

2] concat()

This function joins two or more than two strings and also we can use this function to add two columns.

Syntax : CONCAT(str1,str2,...)
Example : SELECT CONCAT('w3resource', '.', 'com');
Output : w3resource.com

3]group_concat(column_name)
Ex- group_concat(resources_used)

4] SUBSTR()

MySQL SUBSTR() returns the specified number of characters from a particular position of a given string. SUBSTR() is a synonym for SUBSTRING().

Syntax : SUBSTR(str,pos,len)
Example : SELECT SUBSTR('w3resource',4,3);
Output : eso

5] REPLACE()

This function replaces all the occurrences of a substring within a string.

Syntax : REPLACE(str,from_str,to_str)
Example : SELECT REPLACE('w3resource','ur','r');
Output : w3resource

7] REVERSE()

Returns a given string with the order of the characters reversed.

Syntax : REVERSE(str)
Example : SELECT REVERSE('w3resource');
Output : ecruoser3w

8] CAST() – convert integer to char or char to integer
Ex-We use unsigned to convert char to a positive number
1] select Store, concat('Q',cast(10-sum(cast(right(Quarter,1) as unsigned)) as char)) as quarter
2] cast(rank() over(partition by city order by days asc) as signed) - cast(rank() over(partition by city order by cases asc) as signed) as cs_rnk

9] RIGHT(string,1) & LEF(string,n)
N - no of characters
These functions used to extract char from the string using right side or left side
Ex- string = 10$
Output = left(2) = 10 - it giving us 2 char from left side.

10] lower() or higher()
Converts string into lower or upper case

Date Functions:

1] DATE_FORMAT(date,'%Y-%m-%d'):

Ex- date_format('2008-07-06-00-0-00', '%Y-%m-%d')
Output:- 2008-07-06

2] MONTH(date)

Ex- month(2008-07-06)
Output:- 07

3] YEAR(date)

Ex-year(2008-05-12)
Output: 05

4] DAY(date)

5] DATE_DIFF(date2,date1)

Ex- date_diff(2008-05-25,2008-05-22) = 2

EXTRACT(): Returns a single part of a date/time. Syntax:

EX- SELECT Extract(YEAR FROM BirthTime) AS BirthYear

We can give any unit like Day,Month,second,minute,week

DATE_ADD() : Adds a specified time interval to a date

Syntax:

DATE_ADD(date, INTERVAL expr type);

Where, date – valid date expression and expr is the number of intervals we want to add.
and type can be one of the following:
MICROSECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, YEAR, etc.

```
EX-SELECT DATE_ADD(BirthTime, INTERVAL 1 YEAR) AS
   FROM Test;
```

Example for adding one day in the date:-- date_add(strt_date,interval 1 day)

Period_dff(): To find the difference between months of two dates.
period_diff(date_format(now(), '%Y%m'), date_format(time, '%Y%m'))

# Notes:

What is difference between count(*) & count(1) & count(-1) & count(column_name)?
count(*) and count(1) and count(-1) and count('abc') all these will give the same result which is

Count of records in a table including Null values.
So actually what count does is it assigns the value passed in () to each row and counts that value.

count(column_name) just count the no of records without Null values.

# String-based questions:

1] find the occurrence of space in the name.

select name,replace(name,' ','') as new_name, CHAR_LENGTH(name) -
CHAR_LENGTH(replace(name,' ','')) as count_of_spaces
from strings.