# Basics

## Transactions

<https://www.geeksforgeeks.org/sql-transactions/>

# Experiment 2

SQL Queries:

a. Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym, different constraints etc.

b. Write at least 10 SQL queries on the suitable database application using SQL DML statements.

Note: Instructor will design the queries which demonstrate the use of concepts like Insert, Select, Update, Delete with operators, functions, and set operator etc.

## CRUD

<https://www.geeksforgeeks.org/crud-operations-in-mysql/>

## Views

<https://www.w3schools.com/sql/sql_view.asp>

Rules for a view to be updatable:

1. The view must be based on a single table (no joins or unions).
2. Each column in the view must map directly to a column in the base table (no computed or derived columns).
3. The view cannot use aggregate functions like SUM, COUNT, AVG, etc.
4. The view cannot contain GROUP BY clauses.
5. The view cannot include DISTINCT, TOP, or LIMIT clauses.
6. The view cannot have subqueries in the SELECT clause.
7. The base table should ideally have a primary key or unique key.
8. All columns in the view should adhere to NOT NULL constraints in the base table.

Also, updating an updatable view **does** update the underlying table directly, but only those rows that are in the view. If you update the underlying table, the view will be affected.

Updating a view: <https://www.tutorialspoint.com/sql/sql-update-view.htm>

Note: Updates cannot be performed on columns based on which the view was created.

## Indexes

Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries.

<https://www.w3schools.com/sql/sql_create_index.asp>

Even if you haven't created any user-defined indexes on a table, there can still be built-in or automatically generated indexes. Here are the primary types that might exist:

**1. Primary Key Index**

* When you define a **primary key** on a table, most SQL databases automatically create a unique index on that column to enforce the primary key constraint.
* This index ensures that each value in the primary key column is unique and helps the database quickly locate rows based on the primary key.

**2. Unique Constraints**

* If you specify a **unique constraint** on a column, the database will automatically create a unique index to enforce this constraint, ensuring no duplicate values in that column.
* For example:

sql

Copy code

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

email VARCHAR(255) UNIQUE

);

* Here, employee\_id will have an index for the primary key, and email will also have an index due to the unique constraint.

**3. Foreign Key Indexes (Sometimes)**

* In some databases, if you define a **foreign key** on a column, the database may automatically create an index on that column to improve the performance of join operations.
* Note: Not all database systems create an index on foreign keys automatically, so this depends on the system (e.g., MySQL doesn’t do this by default).

**4. Internal System or Metadata Indexes**

* Some databases have hidden or internal indexes for managing specific internal operations, such as row ID indexing, which may not be visible to users but help with row retrieval.
* For example, in Oracle databases, each table has an internal **ROWID** pseudo-column that helps locate rows quickly.

**Checking for Existing Indexes**

You can verify existing indexes on your table by using SHOW INDEX in MySQL or the equivalent command in other SQL systems. It will display both user-defined and automatically generated indexes.

## Sequences

<https://www.geeksforgeeks.org/sql-sequences/>

You can change the sequence too, by

ALTER SEQUENCE employee\_seq

INCREMENT BY 2; -- Changes the increment for future values

To set a sequence to any column, say primary key, you can create sequence first and then

-- Create two sequences

CREATE SEQUENCE emp\_id\_seq

START WITH 1

INCREMENT BY 1;

CREATE SEQUENCE dept\_id\_seq

START WITH 100

INCREMENT BY 1;

-- Create a table that uses both sequences

CREATE TABLE employees (

employee\_id INT PRIMARY KEY DEFAULT nextval('emp\_id\_seq'),

department\_id INT DEFAULT nextval('dept\_id\_seq'),

first\_name VARCHAR(50),

last\_name VARCHAR(50)

);

If instead of creating a custom sequence, you want the autoincrement to start from a specified value, you can do

CREATE TABLE employees (

employee\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50)

) AUTO\_INCREMENT = 1000; -- Set starting point

## Synonym

In SQL, the term **synonym** refers to an alias or alternative name for a database object. This can be used to simplify access to objects or to abstract the underlying structure from users.

**Characteristics of Synonyms:**

* **Alias for Database Objects**: A synonym can point to tables, views, sequences, stored procedures, and other database objects.
* **Easy Access**: Synonyms provide a way to simplify access to database objects, especially when dealing with long object names or complex schemas.
* **User Privileges**: You can grant permissions on a synonym, allowing users to access the underlying object without needing direct permissions on the object itself.

**Example of Creating a Synonym:**

Here’s how to create a synonym for the previously defined sequence:

CREATE SYNONYM emp\_seq FOR employee\_seq;

The general syntax for dropping a sequence is

DROP SYNONYM [ IF EXISTS ] [ schema. ] synonym\_name

The ones we create using are temporary aliases and not synonyms

# Experiment 3

# MongoDB

<https://www.mongodb.com/docs/manual/crud/>