D.Y. Patil Academic Education Excellence Federation's
Dr.D.Y. Patil Technical Campus
(Engineering and MCA)
(Approved by AICTE, DTE-Govt of Maharashtra, Affiliated to Savitribai Phule Pune University, Pune)
Sr.No.32/1/A/7, Near Talegaon Railway Station, A/P Varale, Talegaon Dabhade
Tal-Maval, Dist-Pune 410507
Ph. No.9920141406,9309228311,7666829653,9307909501
Website: www.dypatiltcs.com

Lab Manual – SQL Queries

Experiment No. 2

Title: SQL DDL & DML Statements for Real-Time Database Application

---

Aim :

To design and implement **SQL DDL (Data Definition Language)** statements for creating tables, views, indexes, sequences, synonyms, and constraints, and to perform **SQL DML (Data Manipulation Language)** queries such as insert, update, delete, and select with operators, functions, and set operations.

Problem Statement :

*Develop the SQL schema for the **Online Bookstore Application** (or selected case study) using SQL objects. Perform at least 10 SQL queries that demonstrate different operations like Insert, Select, Update, Delete, Functions, Joins, and Set Operators.*

Theory

SQL Objects

- **Table:** Stores data in rows and columns.
- **View:** Logical representation of data from one or more tables.
- **Index:** Improves query performance by providing fast access.
- **Sequence:** Generates unique numeric values, often for primary keys.
- **Synonym:** Alternate name for database objects.
- **Constraints:** Rules enforced on data.
    - **NOT NULL** – Ensures value is not null.

- **UNIQUE** – Ensures unique values.
- **PRIMARY KEY** – Uniquely identifies each record.
- **FOREIGN KEY** – Establishes relationship between tables.
- **CHECK** – Ensures values meet a condition.
- **DEFAULT** – Provides default value.

SQL Statements

- **DDL (Data Definition Language):** CREATE, ALTER, DROP, TRUNCATE.
- **DML (Data Manipulation Language):** INSERT, UPDATE, DELETE, SELECT.
- **TCL (Transaction Control Language):** COMMIT, ROLLBACK, SAVEPOINT.
- **DCL (Data Control Language):** GRANT, REVOKE.

Tools Used

- Oracle / MySQL / PostgreSQL / SQL Server
- SQL command-line interface or GUI tools (e.g., MySQL Workbench, SQL*Plus, pgAdmin).

Procedure

1. Create required **tables** based on normalized schema.
2. Apply appropriate **constraints** while creating tables.
3. Create **views, indexes, sequences, and synonyms**.
4. Perform **DML operations**: insert records, update records, delete unwanted records.
5. Execute **SQL queries** using operators, functions, joins, and set operators.
6. Verify query results and interpret outputs.

Equations / Rules Applied

- **Primary Key Constraint:** Ensures entity integrity.
- **Foreign Key Constraint:** Ensures referential integrity.

- **Functional Operators:** =, >, <, LIKE, BETWEEN, IN, AND, OR.
- **Aggregate Functions:** COUNT, SUM, AVG, MAX, MIN.
- **Set Operators:** UNION, INTERSECT, MINUS.

---

7. Sample SQL Statements

a. DDL Statements (Bookstore Example)

```
-- Create table with constraints

CREATE TABLE Customer (
  Cust_ID INT PRIMARY KEY,
  Name VARCHAR(50) NOT NULL,
  Email VARCHAR(50) UNIQUE,
  Phone VARCHAR(15)
);


CREATE TABLE Book (
  ISBN VARCHAR(20) PRIMARY KEY,
  Title VARCHAR(100),
  Author VARCHAR(50),
  Price DECIMAL(10,2) CHECK (Price > 0)
);


CREATE TABLE Orders (
  Order_ID INT PRIMARY KEY,
  Order_Date DATE DEFAULT CURRENT_DATE,
  Cust_ID INT,
  FOREIGN KEY (Cust_ID) REFERENCES Customer(Cust_ID)
);


CREATE TABLE Order_Details (
  Order_ID INT,
  ISBN VARCHAR(20),
  Quantity INT CHECK (Quantity > 0),
  PRIMARY KEY (Order_ID, ISBN),
```

```
  FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID),
  FOREIGN KEY (ISBN) REFERENCES Book(ISBN)
);

CREATE TABLE Payment (
  Pay_ID INT PRIMARY KEY,
  Pay_Type VARCHAR(20),
  Amount DECIMAL(10,2),
  Order_ID INT,
  FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID)
);

-- Create View

CREATE VIEW OrderSummary AS
SELECT c.Name, o.Order_ID, o.Order_Date, SUM(b.Price * od.Quantity) AS TotalAmount
FROM Customer c
JOIN Orders o ON c.Cust_ID = o.Cust_ID
JOIN Order_Details od ON o.Order_ID = od.Order_ID
JOIN Book b ON od.ISBN = b.ISBN
GROUP BY c.Name, o.Order_ID, o.Order_Date;

-- Create Index
CREATE INDEX idx_book_title ON Book(Title);

-- Create Sequence
CREATE SEQUENCE seq_order START WITH 1000 INCREMENT BY 1;

-- Create Synonym (Oracle specific)
CREATE SYNONYM Cust FOR Customer;
```

---

b. DML Queries

```
-- 1. Insert records
INSERT INTO Customer VALUES (1, 'Ramesh Sharma', 'ramesh@gmail.com', '9876543210');
```

```
INSERT INTO Book VALUES ('B101', 'DBMS Concepts', 'Korth', 550.00);

-- 2. Update customer details
UPDATE Customer SET Phone = '9999999999' WHERE Cust_ID = 1;

-- 3. Delete a book record
DELETE FROM Book WHERE ISBN = 'B101';

-- 4. Select all customers
SELECT * FROM Customer;

-- 5. Select books with price > 500
SELECT Title, Price FROM Book WHERE Price > 500;

-- 6. Use aggregate functions
SELECT COUNT(*) AS Total_Customers FROM Customer;

-- 7. Join query: Orders with customer names
SELECT c.Name, o.Order_ID, o.Order_Date
FROM Customer c JOIN Orders o ON c.Cust_ID = o.Cust_ID;

-- 8. Set operator: Books priced above 1000 OR authored by 'Korth'
(SELECT Title FROM Book WHERE Price > 1000)
UNION
(SELECT Title FROM Book WHERE Author = 'Korth');

-- 9. Order by price
SELECT Title, Price FROM Book ORDER BY Price DESC;

-- 10. Using LIKE and BETWEEN
SELECT Title FROM Book WHERE Title LIKE '%SQL%' AND Price BETWEEN 300 AND 800;
```

---

8. Results

- Table created successfully messages.
- Records inserted, updated, deleted.
- Result sets showing queried data.
- View displaying summarized order details.

---

9. Conclusion

In this experiment, we implemented **SQL DDL statements** to create tables, views, indexes, sequences, and synonyms with various constraints, and executed **SQL DML queries** to manipulate and retrieve data. This demonstrates the power of SQL in defining database structure and performing data operations efficiently.
