# ***WORKSHEET 6 SQL ANSWER***

1. Which of the following are TCL commands?

**Ans**. Commit, Rollback, Savepoint

2. Which of the following are DDL commands?

**Ans**. Create, Drop, and Alter

3. Which of the following is a legal expression in SQL?

**Ans**. SELECT * FROM SALES WHEN PRICE = NULL;

4. DCL provides commands to perform actions like-

**Ans**. Authorization, Access and other control over the database

5. Which of the following should be enclosed in double quotes?

**Ans**. All of the mentioned

6. Which of the following command makes the updates performed by the transaction permanent in the database?

**Ans**. COMMIT

7. A subquery in an SQL Select statement is enclosed in:

**Ans**. Parenthesis - (...).

8. The result of a SQL SELECT statement is a :-

**Ans**. TABLE

9. Which of the following do you need to consider when you make a table in a SQL?

**Ans**. All of the mentioned

10. If you don't specify ASC and DESC after a SQL ORDER BY clause, the following is used by___?

**Ans**. ASC

11. What is denormalization?

Ans. **Denormalization** is the process of intentionally adding redundancy to a database table in order to improve the read performance of database queries. In other words, denormalization involves adding duplicate data or grouping data in a way that is not normalized to reduce the number of joins required to retrieve data from the database. While normalization is used to minimize redundancy and maintain data consistency, denormalization is used to improve database performance by allowing for faster data retrieval. However, denormalization can also make data updates and maintenance more complex and can lead to inconsistencies if not properly managed.

Denormalization can help optimize query performance for large-scale, complex data systems, but it also increases the risk of data inconsistencies and requires careful management to ensure data accuracy and integrity.

**Denormalization pros:**

- Faster reads for denormalized data
- Simpler queries for application developers
- Less compute on read operations

**Denormalization cons:**

- Slower write operations
- Additional database complexity
- Potential for data inconsistency
- Additional storage required for redundant tables

12. What is a database cursor?

**Ans**. A database cursor is a control structure that enables traversal over the records or rows in a database result set. It is a pointer or a reference to a specific row within a result set, which allows you to move forward or backward through the records and access individual rows or sets of rows as needed. Cursors can be used to update, delete, and retrieve data from a database, and are particularly useful when working with large data sets that would be impractical to load into memory all at once.

Cursors are used to store Database Tables. There are 2 types of Cursors: Implicit Cursors, and Explicit Cursors. These are explained as following below.

**Implicit Cursors:**

Implicit Cursors are also known as Default Cursors of SQL SERVER. These Cursors are allocated by SQL SERVER when the user performs DML operations.

**Explicit Cursors**

Explicit Cursors are Created by Users whenever the user requires them. Explicit Cursors are used for Fetching data from Table in Row-By-Row Manner.

13. What are the different types of the queries?

**Ans**. There are several types of queries that can be used in a database management system:

- Select query: Used to retrieve data from one or more tables.
- Insert query: Used to add new records to a table.
- Update query: Used to modify existing records in a table.
- Delete query: Used to remove records from a table.
- Create query: Used to create new tables, views, or stored procedures in a database.
- Alter query: Used to modify the structure of an existing table or view.
- Drop query: Used to delete tables, views, or stored procedures from a database.
- Join query: Used to combine data from two or more tables based on a common field.
- Subquery: A query within a query used to retrieve data that will be used in the main query.

14. Define constraint?

**Ans**. A constraint is a rule or limitation applied to a database table column or set of columns. Constraints are used to ensure data integrity and consistency by enforcing specific rules or limits on the data that can be entered into a table.

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 - Prevents actions that would destroy links between tables
- CHECK - Ensures that the values in a column satisfy a specific condition
- DEFAULT - Sets a default value for a column if no value is specified
- CREATE INDEX - Used to create and retrieve data from the database very quickly

Each of these constraints serves a different purpose and can be used to ensure that the data in a database is accurate and consistent.

15. What is auto increment?

**Ans**. Auto increment is a feature in databases that allows a unique number to be automatically generated and assigned to a field when a new record is inserted into a table. It is commonly used for primary keys, which are unique identifiers for each record in a table. With auto increment, the database system handles the task of generating and assigning new values, which can simplify the process of inserting new records into the table.

*Setup for SQL Server:*

The SQL Server uses the IDENTITY keyword to set auto increment with the following syntax while creating a table:

IDENTITY (initial_value, increment_value);

In the above syntax:

initial_value: It is the value from where you want the numbering to begin

increment_value: It is the increment between each auto-generated number

Let’s create the same Students table in the SQL Server and use the IDENTITY keyword to set auto increment. This time you will directly start with 1000 and have an interval of 5 between each row’s number.

-- *Creating table*

CREATE TABLE Students( ID int IDENTITY(1000, 5) PRIMARY KEY, FirstName varchar(25) NOT NULL, LastName varchar(25), Age int);

-- *Inserting values*

INSERT INTO Students (FirstName, LastName, Age) VALUES ('Rahul', 'Kumar', 24);

INSERT INTO Students (FirstName, LastName, Age) VALUES ('Aakash', 'Roy', 25);

```
INSERT INTO Students (FirstName, LastName, Age) VALUES ('Nick', 'Chopra', 23);
```

*-- Fetching values*

```
SELECT * FROM Students;
```

*Output*:

| **ID** | **FirstName** | **LastName** | **Age** |
|---|---|---|---|
| 1000 | Rahul | Kumar | 24 |
| 1005 | Aakash | Roy | 25 |
| 1010 | Nick | Chopra | 23 |