1. **SQL Commands**

**Instructions:**

Please share your answers filled in line in the Word document. Submit code separately wherever applicable.

Please ensure you update all the details:

**Name: NISHANTH V Batch ID: CDA 15032023**

**Topic: SQL Commands**

**Assignments: -**

1. What is SQL, and what are its common uses in database management?

SQL stands for Structured Query Language. It is a programming language used for managing and manipulating relational databases. SQL provides a standardized way to interact with databases, allowing users to create, modify, and retrieve data stored in a structured format.

Some common uses of SQL in database management include:

1. Data Definition Language (DDL): SQL is used to define the structure and schema of a database. DDL statements like CREATE, ALTER, and DROP are used to create tables, define relationships between tables, modify table structures, and delete tables.

2. Data Manipulation Language (DML): SQL allows users to retrieve, insert, update, and delete data in the database. DML statements like SELECT, INSERT, UPDATE, and DELETE are used to perform these operations on the data stored in the tables.

3. Data Control Language (DCL): SQL includes commands for controlling access and permissions to the database. DCL statements like GRANT and REVOKE are used to grant or revoke privileges to users, defining their level of access to the database objects.

4. Data Query Language (DQL): SQL provides powerful querying capabilities to retrieve specific data from the database. The SELECT statement is the most commonly used DQL statement, allowing users to specify the columns, filters, and sorting criteria to retrieve the desired data.

5. Data Administration: SQL can be used for administrative tasks such as creating and managing database users, creating backups and restoring data, optimizing performance through indexes and query optimization, and managing transactions.

6. Reporting and Analysis: SQL can be used to extract data from databases and generate reports. It is often used in conjunction with other tools and languages for data analysis and business intelligence purposes.

Overall, SQL is a versatile language that enables efficient management and manipulation of relational databases, making it a fundamental tool in the field of database management.

1. What is a foreign key in SQL, and how is it used to establish relationships between tables?

In SQL, a foreign key is a column or a set of columns in a table that references the primary key of another table. It is used to establish relationships between tables, enabling data integrity and enforcing referential integrity constraints.

Here's how it works:

1. Primary Key: A primary key is a unique identifier for each record in a table. It ensures that each row in the table is uniquely identified. Typically, the primary key is a single column, but it can also be a combination of columns.

2. Foreign Key: A foreign key in one table references the primary key of another table. It establishes a link between the two tables based on a common column or columns. The foreign key column(s) in the referencing table hold values that match the values in the primary key column(s) of the referenced table.

3. Relationship Types: There are different types of relationships that can be established using foreign keys:

- One-to-One (1:1): Each row in one table is related to at most one row in another table.

- One-to-Many (1:N): Each row in one table can be related to multiple rows in another table, but each row in the second table can only be related to one row in the first table.

- Many-to-One (N:1): Each row in one table can be related to at most one row in another table, but each row in the second table can be related to multiple rows in the first table.

- Many-to-Many (N:N): Each row in one table can be related to multiple rows in another table, and vice versa. This relationship requires an intermediate table with foreign keys referencing the primary keys of both tables.

4. Referential Integrity: By defining foreign keys, you can enforce referential integrity. This means that the values in the foreign key column(s) of the referencing table must exist as primary key values in the referenced table. It ensures that relationships between tables are valid and consistent.

5. Cascading Actions: You can also define cascading actions for foreign keys, such as ON DELETE and ON UPDATE. These actions specify what should happen to the related records in the referencing table when a referenced record is deleted or updated. For example, you can choose to cascade the deletion or update to maintain data consistency automatically.

In summary, foreign keys in SQL are used to establish relationships between tables. They enable you to link data between tables, enforce referential integrity, and define various types of relationships.

**DATABASE CREATION: -**

1. Create a database ‘classroom.’

**CREATE DATABASE classroom;**

1. Create a table named ‘science\_class’ with the following properties.

3 columns (enrollment\_no int, name varchar, science\_marks int)

**CREATE TABLE science\_class(**

**enrollment\_no int,**

**name varchar,**

**science\_marks int);**

**INSERTING & IMPORTING: -**

1. Insert the following data into science\_class using insert into command.

| 1 | popeye | 33 |
| --- | --- | --- |
| 2 | olive | 54 |
| 3 | brutus | 98 |

**Insert into science\_class(id, name, age)**

**values (1, ‘popeye’, 33), (2,’olive’,54), (3, ‘brutus’, 98);**

1. Import data from CSV file ‘student.csv’ attached in resources to science\_class to insert data of the next 8 students.

**COPY science\_class FROM ‘path/students.csv’ CSV HEADER LIMIT 8;’**

**ALSO**

**.csv file can be imported in mysql workbench -> server -> data import.**

**SELECT & WHERE: -**

1. Retrieve all data from the table ‘Science\_Class’.

**SELECT \* ,**

**FROM Science\_Class;**

1. Retrieve the name of students who have scored more than 60 marks.

**SELECT name,**

**FROM Science\_Class,**

**WHERE Marks > 60;**

1. Retrieve all data of students who have scored more than 35 but less than 60 marks.

**SELECT \* ,**

**FROM students ,**

**WHERE marks BETWEEN 35 AND 60;**

1. Retrieve all other students i.e., who have scored less than or equal to 35 or more than or equal to 60.

**SELECT \* ,**

**FROM students ,**

**WHERE marks <= 35 OR marks>= 60;**

**UPDATING TABLES: -**

1. Update the marks of ‘popeye’ to 45.

**UPDATE science\_class**

**SET science\_marks= 45**

**WHERE name = 'Popeye';**

1. Delete the row containing details of the student named ‘robb.’

**DELETE from science\_class**

**WHERE name = 'Robb';**

1. Rename column ‘name’ to ‘student\_name’.

**alter table science\_class rename column name to student\_name;**

;