fingerTips

Module-2

Introduction to SQL

1) Structured Query Language (SQL)

SQL stands for Structured Query Language, and it is a programming language that is primarily used for managing and manipulating data within relational database management systems (RDBMS).

SQL provides a standard way to interact with databases, and it allows users to create, read, update, and delete **(CRUD operations)** data from a database. It is widely used in web development, data analysis, and other applications that require working with large amounts of data.

Some of the common operations that can be performed using SQL include creating databases and tables, inserting and updating data, querying data to retrieve specific information, and managing user permissions and security.

2) SQL versus Excel

SQL and Excel are both powerful tools for working with data, but they have different strengths and limitations.

Here are some things that SQL can do that Excel cannot:

i. Work with large datasets: SQL is designed to handle large datasets efficiently and can process millions of records quickly. Excel can become slow and unwieldy when working with large datasets.
ii. Handle complex data relationships: SQL can easily handle complex relationships between tables, which can be difficult or impossible to do in Excel. SQL can join tables, filter data based on multiple conditions, and aggregate data across multiple tables.
iii. Manage data security: SQL provides more advanced security options than Excel. It allows administrators to set up granular permissions to control who can access, modify, or delete data.
iv. Automate data processing: SQL can be used to automate data processing tasks such as data cleaning, data validation, and data transformation. Excel can also automate some

tasks, but it is more limited in this regard.

v. Work collaboratively: SQL databases are designed to support concurrent access by multiple users, making it easier to work collaboratively on a dataset. Excel can be shared, but it can be difficult to manage changes and maintain data integrity.

SQL is a better choice for managing large datasets with complex relationships and managing data security, while Excel is better for ad-hoc data analysis and reporting.

3) What SQL can do?

SQL (Structured Query Language) is a powerful tool for managing and manipulating data within relational database management systems (RDBMS). Here are some of the main abilities of SQL:

i. Data Manipulation: SQL can perform a variety of operations to manipulate data, including adding new data, modifying existing data, and deleting data from a database.

ii. Data Retrieval: SQL can be used to retrieve data from one or more tables in a database using a variety of querying methods, including filtering, sorting, and aggregating data.

iii. Data Definition: SQL can be used to define the structure of a database, including creating tables, defining relationships between tables, and defining constraints to ensure data integrity.

iv. Data Control: SQL provides a range of tools to manage database security, including user authentication, user permissions, and data encryption.

v. Transaction Control: SQL can be used to manage transactions, which are groups of related database operations that must be executed as a single unit.

vi. Reporting: SQL can be used to create customized reports from the data stored in a database, including charts, graphs, and other visualizations.

vii. Integration: SQL can be integrated with other programming languages and tools, such as Python and R, to provide a comprehensive solution for data analysis and manipulation.

SQL is a versatile tool that allows users to manage and manipulate data in a variety of ways, making it an essential tool for anyone who works with data.

4) What are applications of SQL

There are many real-time applications of SQL, including:

i. Banking systems: SQL is used in banking systems to store and manage customer data, account information, and transaction details.
ii. E-commerce websites: SQL is used to store and manage product information, customer data, and sales records.
iii. Healthcare systems: SQL is used to manage patient records, medical history, and appointment schedules.
iv. Logistics and transportation systems: SQL is used to manage inventory, track shipments, and optimize routes.
v. Social media platforms: SQL is used to manage user data, post content, and track engagement metrics.
vi. Online education platforms: SQL is used to manage student records, course materials, and grades.
vii. Travel and hospitality systems: SQL is used to manage hotel bookings, flight schedules, and customer data.
viii. Online gaming platforms: SQL is used to manage user data, game statistics, and leader boards.
ix. Energy management systems: SQL is used to manage energy consumption data, monitor equipment performance, and optimize operations.
x. Retail and sales systems: SQL is used to manage inventory, track sales, and analyze customer behavior.

These are just a few examples of the many real-time applications of SQL. In general, SQL is used in any system that needs to store, manage, and manipulate large amounts of data.

5) Tables & Fields

In SQL (Structured Query Language), a table is a collection of data organized into rows and columns. Each table has a unique name, and each column within the table has a unique name as well.

Tables can be used to store a variety of data types, such as text, numbers, dates, and more.

A field, also known as a column or attribute, is a single piece of data within a table. Each field has a unique name and a specific data type, such as text, number, or date. The data type of a field determines what kind of data can be stored in that field.

For example, let's say you have a table called "Customers". This table have columns "CustomerID", "FirstName", "LastName", "Email", and "Phone". Each of these columns represents a field within the "Customers" table, and each field has a specific data type. "CustomerID" might be an integer, "FirstName" and "LastName" might be text fields, "Email" might be a string, and "Phone" might be a numeric field.

SQL allows you to manipulate and query the data within these tables using commands such as SELECT, INSERT, UPDATE, and DELETE.

6) Keys in SQL

In SQL (Structured Query Language), keys are used to establish relationships between tables and to ensure the integrity of the data. There are several types of keys in SQL, including:

i. Primary key: A primary key is a unique identifier for each row in a table. It is used to ensure that each row is unique and can be accessed quickly. Each table can have only one primary key, and it cannot be null (i.e., it must have a value for each row).

ii. Foreign key: A foreign key is a field in a table that refers to the primary key of another table. It is used to establish relationships between tables and to ensure referential integrity. When a foreign key is defined, it ensures that the data in the referencing table (i.e., the table with the foreign key) is consistent with the data in the referenced table (i.e., the table with the primary key).

iii. Unique key: A unique key is similar to a primary key, but it allows null values. It is used to ensure that a column or

group of columns in a table contains only unique values. Unlike a primary key, a table can have multiple unique keys.

iv. Composite key: A composite key is a key that consists of two or more columns in a table. It is used when a single column cannot uniquely identify each row in a table. In a composite key, the combination of two or more columns must be unique.

v. Candidate key: A candidate key is a set of columns in a table that can uniquely identify each row. It is similar to a primary key, but it has not been selected as the primary key for the table.

vi. Super key: A super key is a set of one or more columns in a table that can uniquely identify each row. It is similar to a candidate key, but it may contain additional columns that are not necessary to uniquely identify each row.

In summary, these different types of keys play an important role in ensuring the integrity and consistency of the data in SQL tables.

7) Data types

In SQL (Structured Query Language), data types are used to specify the type of data that can be stored in a column of a table. The following are some of the common data types available in SQL:

i. Numeric data types: These include integers (INT, SMALLINT, BIGINT), decimal numbers (DECIMAL, NUMERIC), and floating-point numbers (FLOAT, REAL, DOUBLE PRECISION).

ii. Character string data types: These include fixed-length strings (CHAR) and variable-length strings (VARCHAR, TEXT).

iii. Date/time data types: These include DATE, TIME, DATETIME, and TIMESTAMP.

iv. Boolean data type: This is a special data type that can have only two values, TRUE or FALSE.

v. Binary data types: These include BLOB (Binary Large Object), which is used to store large amounts of binary data, and BIT, which is used to store a single bit of

information.

vi. Interval data type: This data type is used to store a period of time between two specific points in time.

vii. XML data type: This data type is used to store XML data.

It's important to choose the appropriate data type for each column in a table based on the nature of the data being stored, to ensure data integrity and optimal performance of the database system.

8) Query

In SQL (Structured Query Language), a query is a request for information from a database. Queries are used to retrieve data from one or more tables, to update or delete data, or to perform other operations on the data.

A query in SQL typically consists of one or more SQL statements, which are executed by the database management system to retrieve or modify data. Some common SQL statements used in queries include:

i. SELECT: This statement is used to retrieve data from one or more tables. It can specify which columns to retrieve, as well as any conditions or criteria for filtering the data.

ii. INSERT: This statement is used to insert new data into a table.

iii. UPDATE: This statement is used to update existing data in a table.

iv. DELETE: This statement is used to delete data from a table.

v. JOIN: This statement is used to combine data from two or more tables based on a common column.

vi. GROUP BY: This statement is used to group data by one or more columns and perform aggregate functions (such as SUM or AVG) on the grouped data.

vii. ORDER BY: This statement is used to sort the data in a query result set by one or more columns.

SQL queries can be simple or complex, depending on the nature of the data being retrieved or modified. Queries are an essential

tool for working with databases, as they allow users to extract the data they need and perform operations on it in a flexible and efficient way.

9) Basic structure of Query

The basic structure of a query in SQL (Structured Query Language) consists of a SELECT statement followed by one or more clauses. The SELECT statement specifies the columns to be included in the result set and the tables from which the data should be retrieved. The clauses are used to filter, group, sort, and aggregate the data as needed. Here's an example of a basic SQL query structure:

```
SELECT column1, column2, ...
FROM table1
[JOIN table2 ON condition]
WHERE condition
GROUP BY column1, column2, ...
HAVING condition
ORDER BY column1, column2, ... [ASC|DESC]
```

10) E-commerce dataset introduction

An e-commerce database includes multiple tables to store information about customers, products, and orders. These tables are designed to work together to provide a comprehensive view of ecommerce activity, including customer behavior, product inventory, and order fulfillment. Here is an introduction to the three main tables in an ecommerce database:

i. Customers Table: This table stores information about individual customers who interact with the ecommerce site. It typically includes fields such as customer name, email address, phone number, shipping address, and billing information. This table may also include fields to track customer behavior, such as login history, purchase history, and shopping cart activity.

ii. Products Table: This table stores information about the products available for purchase on the ecommerce site. It

typically includes fields such as product name, description, price, SKU, and inventory level. This table may also include fields to track product characteristics, such as size, color, and weight.

iii. Orders Table: This table stores information about individual orders placed by customers on the ecommerce site. It typically includes fields such as order number, order date, customer ID, product ID, quantity, price, and shipping information. This table may also include fields to track order status, such as order confirmation, payment processing, and shipment tracking.

These three tables work together to provide a complete picture of ecommerce activity, from the individual customers who make purchases to the products they buy and the orders they place. By organizing data in this way, an ecommerce database can help businesses track sales, monitor inventory levels, and optimize the customer experience.

11) Questions which can be answered using SQL

Imagine yourself as the owner of Fashionworld, head of Finance, Products, or Operations. What would you like to know about your business?

- How much are we selling daily? Is it high or low compared to previous days/week/month/year?
- What are the purchasing behavior of our customers? Are they buying more or less?
- What are we selling the most and the least? What are we making money on? Are some products/categories selling more to a particular group of customers?
- What geographic location are we doing well/not well?
- What marketing channel are we doing well on?

12) Create Database

create schema fashionworld;

13) Create tables

Query to create table products:

```
CREATE TABLE products (
  id INT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  price DECIMAL(10,2) NOT NULL,
  size VARCHAR(10),
  color VARCHAR(20),
  description VARCHAR(250)
);
```

Query to create table customers:

```
CREATE TABLE customers (
  id INT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255) NOT NULL,
  phone VARCHAR(20),
  address VARCHAR(255)
);
```

Query to create table orders:

```
CREATE TABLE orders (
  id INT PRIMARY KEY,
```

```
  customer_id INT NOT NULL,
  product_id INT NOT NULL,
  quantity INT NOT NULL,
  order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (customer_id) REFERENCES customers(id),
  FOREIGN KEY (product_id) REFERENCES products(id)
);
```

14) Insert values into tables

To insert values into table products:

```
INSERT INTO products (id, name, price, size, color, description)
VALUES
(1, 'T-shirt', 19.99, 'M', 'Blue', 'A comfortable and stylish t-shirt'),
(2, 'Jeans', 49.99, '32x34', 'Black', 'A classic pair of black jeans'),
(3, 'Sneakers', 79.99, '10.5', 'White', 'A pair of comfortable and stylish sneakers'),
(4, 'Sweater', 34.99, 'L', 'Gray', 'A cozy and warm sweater'),
(5, 'Dress', 59.99, 'S', 'Red', 'A beautiful and elegant dress'),
(6, 'Jacket', 99.99, 'XL', 'Green', 'A warm and stylish jacket'),
(7, 'Skirt', 29.99, 'M', 'Yellow', 'A cute and flirty skirt'),
(8, 'Blouse', 39.99, 'L', 'Pink', 'A flowy and feminine blouse'),
(9, 'Shorts', 24.99, 'S', 'Orange', 'A comfortable pair of shorts for summer'),
(10, 'Hoodie', 49.99, 'L', 'Black', 'A cozy and casual hoodie'),
(11, 'Boots', 89.99, '9.5', 'Brown', 'A stylish pair of boots for any occasion'),
(12, 'Sweatpants', 29.99, 'M', 'Gray', 'A comfortable and casual pair of sweatpants'),
(13, 'Sunglasses', 19.99, NULL, 'Black', 'A cool and trendy pair of sunglasses'),
(14, 'Scarf', 14.99, NULL, 'Purple', 'A warm and cozy scarf for the winter'),
```

(15, 'Hat', 9.99, 'One size', 'Navy', 'A stylish and versatile hat for any outfit'),
(16, 'Jumpsuit', 69.99, 'M', 'Black', 'A chic and trendy jumpsuit for any occasion'),
(17, 'Blazer', 79.99, 'L', 'White', 'A sophisticated and stylish blazer for work or events'),
(18, 'Sweatshirt', 39.99, 'XL', 'Pink', 'A comfortable and cozy sweatshirt for lounging'),
(19, 'Leggings', 24.99, 'S', 'Black', 'A versatile and comfortable pair of leggings'),
(20, 'Pants', 54.99, '32x30', 'Khaki', 'A classic and stylish pair of khaki pants');

To insert values into table customers:

INSERT INTO customers (id, name, email, phone, address)
VALUES
(1, 'John Smith', 'john.smith@example.com', '+1 555-123-4567', '123 Main St, Anytown, USA'),
(2, 'Jane Doe', 'jane.doe@example.com', '+1 555-987-6543', '456 Maple Ave, Anytown, USA'),
(3, 'Bob Johnson', 'bob.johnson@example.com', NULL, '789 Oak St, Anytown, USA'),
(4, 'Emily Williams', 'emily.williams@example.com', '+1 555-555-1212', '321 Elm St, Anytown, USA'),
(5, 'David Lee', 'david.lee@example.com', '+1 555-555-5555', '567 Pine St, Anytown, USA'),
(6, 'Sarah Kim', 'sarah.kim@example.com', '+1 555-123-7890', '890 Cedar Ave, Anytown, USA'),
(7, 'Michael Chen', 'michael.chen@example.com', '+1 555-999-8888', '246 Birch Blvd, Anytown, USA'),
(8, 'Jessica Brown', 'jessica.brown@example.com', '+1 555-777-6666', '369 Spruce St, Anytown, USA'),
(9, 'Kevin Garcia', 'kevin.garcia@example.com', '+1 555-111-2222', '802 Maplewood Dr, Anytown, USA'),
(10, 'Ashley Davis', 'ashley.davis@example.com', NULL, '135 Walnut St, Anytown, USA');

To insert values into table orders:

```
INSERT INTO orders (id, customer_id, product_id, quantity, order_date)
VALUES
(1, 1, 1, 2, '2022-03-08 14:25:00'),
(2, 2, 1, 1, '2022-03-07 09:32:00'),
(3, 3, 3, 4, '2022-03-06 18:05:00'),
(4, 4, 5, 3, '2022-03-05 10:12:00'),
(5, 5, 2, 2, '2022-03-04 15:22:00'),
(6, 1, 3, 1, '2022-03-03 12:48:00'),
(7, 2, 4, 2, '2022-03-02 17:09:00'),
(8, 3, 1, 3, '2022-03-01 11:35:00'),
(9, 4, 2, 1, '2022-02-28 16:02:00'),
(10, 5, 5, 2, '2022-02-27 13:24:00'),
(11, 1, 2, 3, '2022-02-26 10:49:00'),
(12, 2, 3, 2, '2022-02-25 14:56:00'),
(13, 3, 4, 1, '2022-02-24 09:17:00'),
(14, 4, 1, 2, '2022-02-23 12:40:00'),
(15, 5, 3, 3, '2022-02-22 16:58:00');
```

15) Constraints in SQL

Constraints are rules defined on a table's columns that restrict the type of data that can be inserted or updated in that table. Constraints ensure data integrity and help maintain the accuracy and consistency of the database.

There are several types of constraints that can be defined in SQL:

i. NOT NULL constraint: This ensures that a column cannot contain NULL values. If a user tries to insert or update a row with a NULL value in a NOT NULL column, an error will be raised.

ii. UNIQUE constraint: This ensures that each value in a column is unique. If a user tries to insert a value that already exists in the column, an error will be raised.

iii. PRIMARY KEY constraint: This is a combination of a NOT NULL and UNIQUE constraint. It ensures that each row in a table is uniquely identified by a specific column or

combination of columns.

iv. FOREIGN KEY constraint: This ensures that values in a column match values in another table's column, typically used for maintaining referential integrity.
v. CHECK constraint: This ensures that values in a column meet a specific condition or set of conditions.
vi. DEFAULT constraint: This specifies a default value for a column if no value is specified during an insert.

Constraints help maintain the accuracy and consistency of the database, and prevent data from being added that doesn't conform to the table's defined structure.