**Online Retail Sales Database Design – Detailed Report**

The objective of this project is to design and implement a normalized SQL database schema for an **Online Retail Sales Platform**. This platform aims to efficiently manage products, customers, orders, and payments. The goal is to ensure the system is scalable, consistent, and supports transactional operations and reporting.

To achieve this, tools like **MySQL Workbench** are used for database development and execution, while **dbdiagram.io** helps visualize the schema with an ER diagram. The core language used is SQL, with a focus on Data Definition Language (DDL) and Data Manipulation Language (DML).

The key entities identified in the system are:

* **Customers**, who place orders. Each customer has a unique ID, name, email, phone, and address.
* **Products**, which are sold on the platform. Each product includes an ID, name, description, price, and available stock quantity.
* **Orders**, which record purchases. Each order is linked to a customer, has an order date, and total amount.
* **Order Items**, which break down each order into specific products and quantities purchased.
* **Payments**, which track how and when each order was paid.

The **ER diagram** created in dbdiagram.io shows the relationships between these entities using foreign keys. One customer can place many orders, each order can have multiple order items, and every order has one corresponding payment.

The database is created using the command CREATE DATABASE online\_retail\_db; followed by table definitions using CREATE TABLE statements. Each table includes necessary constraints like PRIMARY KEY, FOREIGN KEY, and UNIQUE to maintain data integrity.

Once the tables are created, **sample data** is inserted. Two customers, three products, two orders, four order items, and two payments are added using INSERT statements. This data helps in testing the design and running meaningful queries.

The schema is **normalized up to 3rd Normal Form (3NF)**:

* **1NF** ensures all fields have atomic (indivisible) values.
* **2NF** ensures all non-key attributes are fully dependent on the entire primary key.
* **3NF** removes transitive dependencies, ensuring that non-key fields depend only on the primary key.

With the data populated, several **SQL queries** are written to demonstrate reporting:

* A query to calculate total units sold and sales per product using JOIN and GROUP BY.
* Another query joins orders with customers and payments to show a complete order summary.
* A SQL **VIEW** called sales\_summary is created for easy access to summarized order information.

In conclusion, this database project offers a solid and normalized design for an online retail system. It effectively handles data management for key operations and provides support for generating reports. This structure ensures accuracy, scalability, and ease of data retrieval, making it suitable for real-world e-commerce platforms.