Week 9 – Database security

Aim of the assignment:

* Set up access control in the BikeCorpDB database system
* Define user types and which users may access which data
* Implement access restrictions
* Presentation of my solution and the reasoning behind the decision made

To-do:

* Analyse the data in the BikeCorpDB tables and determine which columns contain sensitive data (based on the shared GDPR doc)
* Define user roles i.e. types of users that can access the database and what kind of data they are allowed to read or write to
* Implement access control – Through MySQL or through Python, or both(??)
* Implement logging system

Data sensitivity

**Highly Sensitive Data**

* **Personal identifiers**: customers.first\_name, customers.last\_name, customers.email, customers.phone, customers.street, customers.city, customers.state, customers.zip\_code
  + *Reason*: These directly identify a person and could enable identity theft or unwanted contact if breached.
* **Staff personal information**: staffs.first\_name, staffs.last\_name, staffs.email, staffs.phone
  + *Reason*: Employee personal data deserves the same protection as customer data.
* **Manager relationships**: staffs.manager\_id
  + *Reason*: Shows organizational hierarchy which could be sensitive internal information.

**Moderately Sensitive Data**

* **Order details**: orders.customer\_id, orders.order\_date, orders.required\_date, orders.shipped\_date
  + *Reason*: Reveals customer purchasing patterns and delivery addresses.
* **Order items**: order\_items.order\_id, order\_items.product\_id, order\_items.quantity, order\_items.list\_price, order\_items.discount
  + *Reason*: Shows what specific products customers purchased and how much they paid.
* **Employee status**: staffs.active
  + *Reason*: Could indicate information about employment status.

**Less Sensitive Data**

* **Product information**: products.product\_name, products.model\_year, products.list\_price, products.brand\_id, products.category\_id
  + *Reason*: Generally public business information.
* **Store information**: stores.name, stores.street, stores.city, stores.state, stores.zip\_code, stores.phone, stores.email
  + *Reason*: Typically public business contact information.
* **Inventory information**: stocks.quantity
  + *Reason*: Internal business data but not personally identifiable.
* **Brand and category information**: brands.brand\_name, categories.category\_name
  + *Reason*: Public product classification information.

USER ROLES AND ACCESS RIGHTS

**1. Administrator**

* **Description**: System administrators who manage the entire database
* **Access rights**:
  + Full access to all tables (SELECT, INSERT, UPDATE, DELETE)
  + Ability to create/modify user accounts and permissions
* **Reasoning**: Administrators need comprehensive access to maintain the system, troubleshoot issues, and manage user accounts. While they have full access, their activities should be thoroughly logged for accountability.

2. **Executive Admin**

* Represents the top-level executive (Staff ID 1)
* Has visibility across all stores and all staff
* Has full access to sensitive business metrics and staff information
* Can see aggregated sales and inventory data across all stores

3. **Store Manager**

* Represents store-specific managers (Staff IDs 2, 5, and 8)

**Description**: Oversees store operations and staff

**Access rights**:

* **Full access** (SELECT, INSERT, UPDATE) to:
  + products
  + stocks
  + stores (their assigned store only)
* **Read/limited update** access to:
  + staffs (view all staff, update only for staff in their store)
  + orders (view all orders from their store, update status)
* **Read-only** access to:
  + customers (limited columns: customer\_id, first\_name, last\_name, email, phone)
  + order\_items (for orders from their store)
* **No access** to:
  + Customer street addresses
  + Staff personal details outside their store

**Reasoning**: Managers need to oversee their store's operations, manage inventory, and monitor sales performance. They need limited access to customer information to handle customer service issues but don't need full customer address details for most tasks.

**4. Sales Staff**

* **Description**: Handles customer orders and basic service
* **Access rights**:
  + **Full access** (SELECT, INSERT, UPDATE) to:
    - orders (create new orders and update order status)
    - order\_items (add items to orders)
  + **Read-only** access to:
    - products (to see product information)
    - stocks (to check availability)
    - customers (limited columns: customer\_id, first\_name, last\_name, email, phone)
  + **No access** to:
    - Customer address details
    - Staff information
    - Store management information
* **Reasoning**: Sales staff need to process orders and help customers, but they don't need access to customer addresses or detailed staff information. They need to see product availability to inform customers about stock status.

**5. Inventory Staff**

* **Description**: Manages product stock and warehouse operations
* **Access rights**:
  + **Full access** (SELECT, INSERT, UPDATE) to:
    - stocks (update inventory levels)
  + **Read-only** access to:
    - products (to identify products)
    - orders (to fulfill orders)
    - order\_items (to know what products to ship)
  + **No access** to:
    - Customer information
    - Staff details
* **Reasoning**: Inventory staff need to manage stock levels and fulfill orders, but they don't need access to customer or staff personal information.

**6. Customer**

* **Description**: End users who purchase products
* **Access rights**:
  + **Read-only** access to:
    - Their own orders and order items
    - Product information (name, description, price)
    - Store information (for contact purposes)
  + **No access** to:
    - Other customers' data
    - Internal business data (staff, inventory)
* **Reasoning**: Customers should only see their own order history and general product/store information. This respects privacy and prevents data leakage.

**Real-world Application Context:**

Let me explain how this would fit into a real-world application:

Imagine the BikeCorpDB is used by a chain of bike stores. Each store has a physical location with several staff members. The database system would be part of their overall business application that handles:

1. **In-store Point of Sale (POS) systems** - Staff members log in to process sales
2. **Inventory management** - Store managers check and update stock levels
3. **Customer portal** - Customers log in to check order status and purchase history
4. **Executive dashboard** - Regional managers and executives monitor performance

In this real-world scenario, our authentication system would be the gateway to all these functions. For example:

* When a sales associate comes to work in the morning, they would log in to the POS system using their credentials
* The authenticate\_user() function would verify their identity
* Based on their role ("staff"), they would only be allowed to see specific data (customer names and contact info, but not full addresses)
* They could only see orders and inventory for their specific store
* All their actions would be logged for accountability

For the executive using a dashboard:

* They would log in using their executive credentials
* The system would verify they have the "executive" role
* They would see aggregated data across all stores
* They could drill down into specific store performance

The key difference in a real system would be integration with a proper web or desktop application framework (like Django, Flask, or a .NET application) with secure session management, API authentication tokens, and more sophisticated user interfaces.

In our BikeCorpDB project, we're building a simplified version of this access control layer to demonstrate the principles without getting into the full complexity of a production system.

Does that help clarify the context? If you'd like, I can go deeper into any specific aspect of how this would work in a real-world scenario.

**For Your Presentation**

In your presentation, you should cover:

1. **Introduction to Data Security Challenges**
   * Brief overview of the BikeCorpDB schema
   * Explanation of GDPR principles and data sensitivity
2. **Data Categorization**
   * Show how you classified data by sensitivity
   * Explain the reasoning behind your classifications
3. **User Role Design**
   * Introduce the different user roles you defined
   * Explain the principle of least privilege
   * Show what data each role needs access to and why
4. **Implementation Approach**
   * Explain how you implemented security using Python
   * Demonstrate column-level and row-level security
   * Show how access logging works for audit purposes
5. **Security Demonstration**
   * Show examples of:
     + Admin accessing all data
     + Manager accessing only their store's data
     + Sales staff seeing limited customer information
     + Customer accessing only their own orders
     + Attempted unauthorized access being blocked
6. **Reflection and Best Practices**
   * Discuss strengths and limitations of your approach
   * Mention additional security measures that could be implemented
   * Reflect on the balance between security and usability