INVENTORY MANAGEMENT SYSTEM (INTERMEDIATE-LEVEL)

You can learn the concept of CURD operations, joins and views in this level of project i.e. intermediate level of inventory management system.

User Table

- **Insert (Create):**

  INSERT INTO inventory.user (firstName, lastName, username, email, passwordHash)
  VALUES ("John", "Doe", "johndoe", "john.doe@email.com", "hashed_password");

- **Update:**

  UPDATE inventory.user
  SET firstName = "Jane", email = "jane.doe@email.com"
  WHERE id = 1;

- **Delete:**

  DELETE FROM inventory.user
  WHERE id = 2;

- **select:**

  SELECT * FROM `inventory`.`user`;

- **Views(active user)**

  CREATE VIEW `active_users` AS

  SELECT * FROM `inventory`.`user`

  WHERE `lastLogin` IS NOT NULL;

- **Joins(join with order)**

  SELECT u.*, o.*

  FROM `inventory`.`user` u

  JOIN `inventory`.`order` o ON u.id = o.userId;

PRODUCT TABLE:

- **Insert (Create):**

  INSERT INTO `inventory`.`product` (`title`, `summary`, `type`, `createdAt`, `updatedAt`, `content`)

  VALUES ('Product Title', 'Product summary.', 1, NOW(), NULL, 'Product content goes here.');

- **Select:**
  SELECT * FROM `inventory`.`product`;

- **Update:**
  UPDATE `inventory`.`product`

  SET `title` = 'New Title'

  WHERE `id` = 1;

- **Delete:**

  DELETE FROM `inventory`.`product`

  WHERE `id` = 1;

VIEW (product by type)

CREATE VIEW `products_by_type` AS

SELECT * FROM `inventory`.`product`

WHERE `type` = 1;

JOIN (join with product meta)

SELECT p.*, pm.*

FROM `inventory`.`product` p

JOIN `inventory`.`product_meta` pm ON p.id = pm.productId;

PRODUCT META TABLE:

Category tale

- **Insert (Create):**

  INSERT INTO `inventory`.`category` (`parentId`, `title`, `metaTitle`, `slug`, `content`)

  VALUES (NULL, 'Main Category', 'Main Category', 'main-category', 'Main category description.');

- **Update:**

  UPDATE `inventory`.`category`

  SET `title` = 'New Title'

  WHERE `id` = 1;

- **Delete:**

  DELETE FROM `inventory`.`category`

  WHERE `id` = 1;

- **select:**

  SELECT * FROM `inventory`.`category`;

- **Joins(with product category)**

  SELECT c.*, pc.*

  FROM `inventory`.`category` c

  JOIN `inventory`.`product_category` pc ON c.id = pc.categoryId;

Product Category Table:

CRUD Operations:

Same as the Category Table.

Brand Table:

CRUD Operations:

Same as the Product Table.

ORDER TABLE:

- **Insert (Create):**

  INSERT INTO `inventory`.`order` (`userId`, `type`, `status`, `subTotal`, `itemDiscount`, `tax`, `shipping`, `total`, `promo`, `discount`, `grandTotal`, `createdAt`, `updatedAt`, `content`)

  VALUES (1, 1, 1, 100.00, 10.00, 5.00, 5.00, 110.00, 'PROMO123', 10.00, 100.00, NOW(), NULL, 'Order details.');

- **Update:**

  UPDATE `inventory`.`order`

  SET `status` = 2

  WHERE `id` = 1;

- **Delete:**

  DELETE FROM `inventory`.`order`

  WHERE `id` = 1;

- **select:**

  SELECT * FROM `inventory`.`order`;

- **Joins(with user)**

  SELECT o.*, u.*

  FROM `inventory`.`order` o

  JOIN `inventory`.`user` u ON o.userId = u.id;

ADDRESS TABLE:

CRUD Operations:

Same as the User Table.

Views: None.

- **Joins(with orders)**

  SELECT a.*, o.*

  FROM `inventory`.`address` a

  JOIN `inventory`.`order` o ON a.orderId = o.id;

ITEM TABLE:

CRUD Operations:

Same as the Product Table.

Views: None.

- **Joins(with product and brand)**

  SELECT i.*, p.*, b.*

  FROM `inventory`.`item` i

  JOIN `inventory`.`product` p ON i.productId = p.id

  JOIN `inventory`.`brand` b ON i.brandId = b.id;

ORDER ITEM TABLE:

CRUD Operations:

Same as the Product Table.

Views: None.

- **Joins(with product)**

  SELECT oi.*, p.*

  FROM `inventory`.`order_item` oi

  JOIN `inventory`.`product` p ON oi.productId = p.id;

TRANSACTIONN TABLE:

CRUD Operations:

Same as the Product Table.

Views: None.

- **Joins(with order)**

  SELECT t.*, o.*

  FROM `inventory`.`transaction` t

  JOIN `inventory`.`order` o ON t.orderId = o.id;