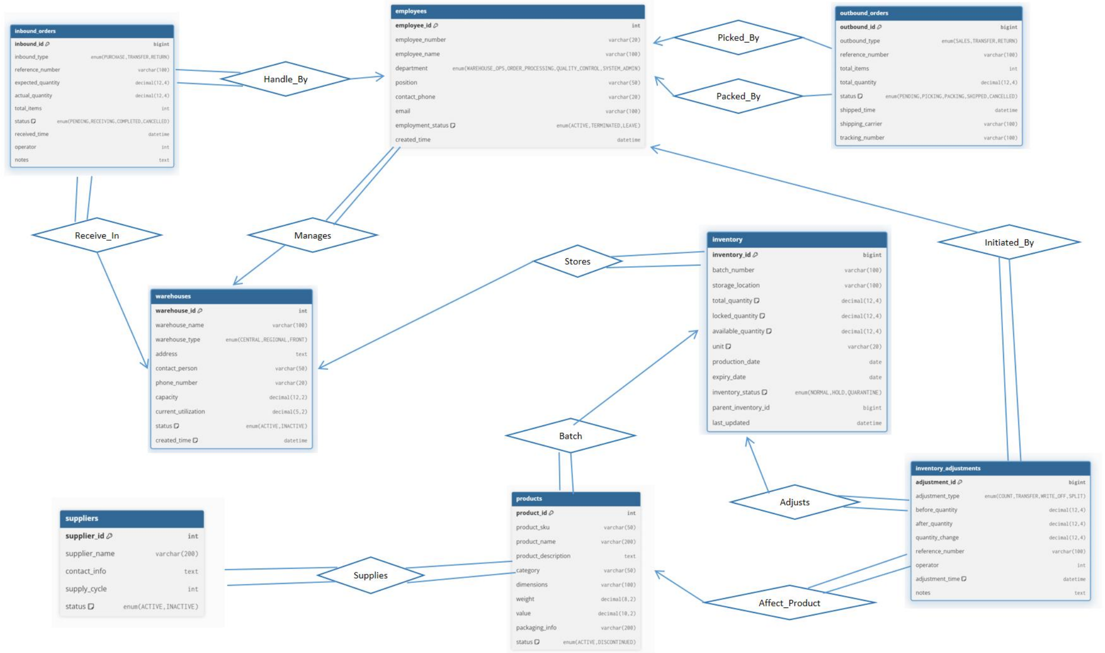
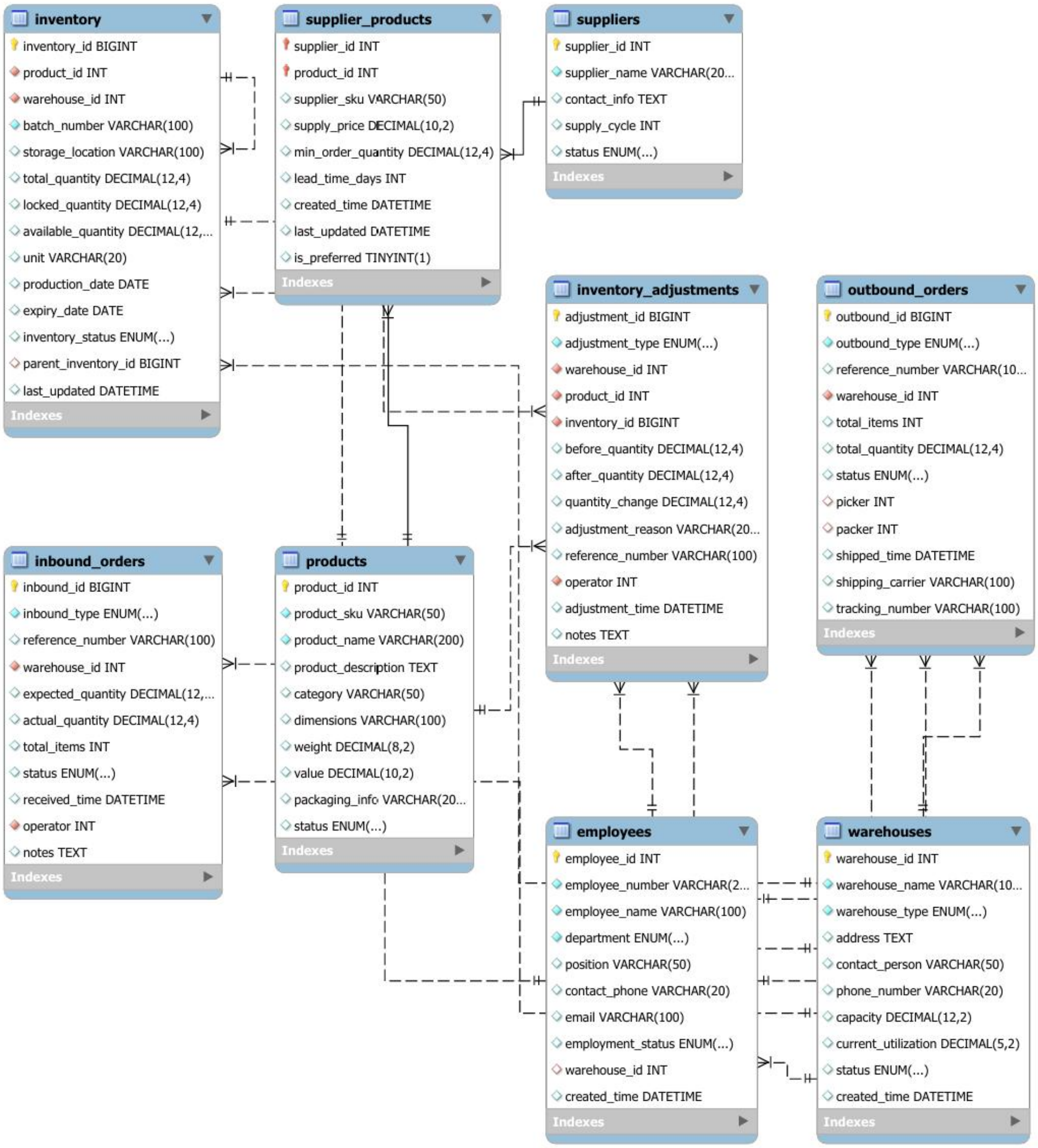
Q1


inbound_orders
inbound_id bigint
inbound_type enum(PURCHASE,TRANSFER,RETURN)
reference_number varchar(100)
expected_quantity decimal(12,4)
actual_quantity decimal(12,4)
total_items int
status enum(PENDING,RECEIVING,COMPLETED,CANCELLED)
received_time datetime
operator int
notes text
Handle_By
employees
employee_id int
employee_number varchar(20)
employee_name varchar(100)
department enum(WAREHOUSE_OPS,ORDER_PROCESSING,QUALITY_CONTROL,SYSTEM_ADMIN)
position varchar(50)
contact_phone varchar(20)
email varchar(100)
employment_status enum(ACTIVE,TERMINATED,LEAVE)
created_time datetime
Picked_By
Packed_By
outbound_orders
outbound_id bigint
outbound_type enum(SALES,TRANSFER,RETURN)
reference_number varchar(100)
total_items int
total_quantity decimal(12,4)
status enum(PENDING,PICKING,PACKING,SHIPPED,CANCELLED)
shipped_time datetime
shipping_carrier varchar(100)
tracking_number varchar(100)
Receive_In
Manages
Stores
Initiated_By
warehouses
warehouse_id int
warehouse_name varchar(100)
warehouse_type enum(CENTRAL,REGIONAL,FRONT)
address text
contact_person varchar(50)
phone_number varchar(20)
capacity decimal(12,2)
current_utilization decimal(5,2)
status enum(ACTIVE,INACTIVE)
created_time datetime
inventory
inventory_id bigint
batch_number varchar(100)
storage_location varchar(100)
total_quantity decimal(12,4)
locked_quantity decimal(12,4)
available_quantity decimal(12,4)
unit varchar(20)
production_date date
expiry_date date
inventory_status enum(NORMAL,HOLD,QUARANTINE)
parent_inventory_id bigint
last_updated datetime
Batch
Adjusts
inventory_adjustments
adjustment_id bigint
adjustment_type enum(COUNT,TRANSFER,WRITE_OFF,SPLIT)
before_quantity decimal(12,4)
after_quantity decimal(12,4)
quantity_change decimal(12,4)
reference_number varchar(100)
operator int
adjustment_time datetime
notes text
suppliers
supplier_id int
supplier_name varchar(200)
contact_info text
supply_cycle int
status enum(ACTIVE,INACTIVE)
Supplies
products
product_id int
product_sku varchar(50)
product_name varchar(200)
product_description text
category varchar(50)
dimensions varchar(100)
weight decimal(8,2)
value decimal(10,2)
packaging_info varchar(200)
status enum(ACTIVE,DISCONTINUED)
Affect_Product

(1) Manages: Warehouse-Employee, 1-N, Employees are affiliated with the warehouse.
(2) Supplies: Supplier-Product, N-N, Suppliers provide multiple products.
(3) Stores: Warehouse-Inventory, 1-N, Warehouse owns the inventory records.
(4) Batch: Product-Inventory, 1-N, One product corresponds to multiple inventory lots.
(5) Split_From: Inventory-Inventory, 1-N, inventory batch splits produce new inventory records.
(6) Receives_In: Warehouse-Inbound_Order, 1-N, Inbound orders belong to a specific warehouse.
(7) Handled_By: Employee-Inbound_Order, 1-N, Employees handle inbound order operations.
(8) Picked_By: Employee-Outbound_Order, 1-N, Employees perform picking for outbound orders.
(9) Packed_By: Employee-Outbound_Order, 1-N, Employees perform packing for outbound orders.
(10) Adjusts: Inventory-Inventory_Adjustment, 1-N, each inventory item can have multiple adjustments.
(11) Initiated_By: Employee-Adjustment, 1-N, adjustments are initiated by employees.
(12) Located_In: Warehouse-Adjustment, 1-N, adjustments occur at a specific warehouse.
(13) Affects_Product: Product-Adjustment, 1-N, adjustments relate to specific products.

Q2.



1. Mapping Entities to Relations:
   Each entity in the ER model is converted into a relation (table). All major entities such as warehouses, products, suppliers, inventory, and orders become tables, and their identifiers become primary keys.
2. Mapping Attributes to Columns:
   All attributes of each entity are mapped to columns in their corresponding tables. Appropriate data types are chosen to store descriptive, numeric, and date-based information.
3. Assigning Primary Keys:
   The unique identifiers defined in the ER model become the primary keys of their tables. These keys ensure that each record is uniquely identifiable and support referential integrity.
4. Mapping One-to-Many Relationships:

For every One-to-Many relationship, a foreign key is added to the table on the "many" sides. This preserves dependencies such as each inventory record belonging to one product and one warehouse.

5. Mapping Many-to-Many Relationships:
   Many-to-Many relationships are transformed into separate bridge tables that contain the primary keys of both related entities. For example, supplier–product relationships become the supplier_products table.
6. Mapping Relationships with Attributes:
   If a relationship has its own attributes, it is converted into a separate table. For example, inventory_adjustments become their own relation to store details like quantity changes and reasons.
7. Handling Weak or Dependent Entities:
   Entities that depend on other entities include the primary key of the owner entity as a foreign key. This preserves necessary dependencies, such as outbound orders referencing inbound orders.
8. Handling Derived Attributes:
   Derived attributes are either omitted or implemented as computed fields. Their presence in the schema diagram clarifies the logic while avoiding redundant storage.
9. Producing the Final Schema Diagram:
   After all mappings are completed, the final schema diagram is created to show tables, attributes, primary keys, foreign keys, and relationship links using crow's-foot notation.