```
-- Database schema for Ship Breaking Management System

-- Machines Table (weights in kg)
CREATE TABLE Machines (
  machine_id INT AUTO_INCREMENT PRIMARY KEY,
  machine_name VARCHAR(100) NOT NULL,
  buying_price DECIMAL(12,2) NOT NULL DEFAULT 0.00,
  buying_date DATE NOT NULL,
  machine_weight_kg DECIMAL(10,2) NOT NULL DEFAULT 0.00,
  bought_from VARCHAR(100) NOT NULL DEFAULT 'Unknown'
);

-- Parts Table
CREATE TABLE Parts (
  part_id INT AUTO_INCREMENT PRIMARY KEY,
  machine_id INT NOT NULL,
  part_name VARCHAR(100) NOT NULL,
  category VARCHAR(50) NOT NULL,
  weight_kg DECIMAL(10,2) NOT NULL CHECK (weight_kg >= 0),
  wastage_weight_kg DECIMAL(10,2) NOT NULL DEFAULT 0.00 CHECK (wastage_weight_kg >= 0),
  min_selling_price DECIMAL(10,2) NOT NULL DEFAULT 0.00 CHECK (min_selling_price >= 0),
  status VARCHAR(50) NOT NULL DEFAULT 'in stock' CHECK (status IN ('in stock', 'sold', 'reserved', 'damaged')),
  CONSTRAINT fk_parts_machine FOREIGN KEY (machine_id) REFERENCES Machines(machine_id) ON DELETE RESTRICT
```

```
);

-- Expenses Table
CREATE TABLE Expenses (
  expense_id INT AUTO_INCREMENT PRIMARY KEY,
  machine_id INT NOT NULL,
  crane_fee DECIMAL(10,2) NOT NULL DEFAULT 0.00 CHECK (crane_fee >= 0),
  delivery_fee DECIMAL(10,2) NOT NULL DEFAULT 0.00 CHECK (delivery_fee >= 0),
  labor_fee DECIMAL(10,2) NOT NULL DEFAULT 0.00 CHECK (labor_fee >= 0),
  broker_fee DECIMAL(10,2) NOT NULL DEFAULT 0.00 CHECK (broker_fee >= 0),
  wastage_cost DECIMAL(10,2) NOT NULL DEFAULT 0.00 CHECK (wastage_cost >= 0),
  expense_date DATE NOT NULL,
  notes VARCHAR(255) NOT NULL DEFAULT '',
  CONSTRAINT fk_expenses_machine FOREIGN KEY (machine_id) REFERENCES
Machines(machine_id) ON DELETE RESTRICT
);

-- SalesEvents Table
CREATE TABLE SalesEvents (
  sale_id INT AUTO_INCREMENT PRIMARY KEY,
  buyer_name VARCHAR(100) NOT NULL,
  sale_date DATE NOT NULL,
  notes VARCHAR(255) NOT NULL DEFAULT ''
);

-- PartSales Table
```

```
CREATE TABLE PartSales (
  part_sale_id INT AUTO_INCREMENT PRIMARY KEY,
  part_id INT NOT NULL,
  sale_id INT NOT NULL,
  quantity INT NOT NULL DEFAULT 1 CHECK (quantity >= 1),
  selling_price DECIMAL(10,2) NOT NULL DEFAULT 0.00 CHECK (selling_price >= 0),
  melting_price DECIMAL(10,2) NOT NULL DEFAULT 0.00 CHECK (melting_price >= 0),
  CONSTRAINT fk_partsales_part FOREIGN KEY (part_id) REFERENCES Parts(part_id) ON DELETE RESTRICT,
  CONSTRAINT fk_partsales_sale FOREIGN KEY (sale_id) REFERENCES SalesEvents(sale_id) ON DELETE RESTRICT
);

-- Indexes for optimization
CREATE INDEX idx_parts_machine_id ON Parts(machine_id);
CREATE INDEX idx_expenses_machine_id ON Expenses(machine_id);
CREATE INDEX idx_salesevents_sale_date ON SalesEvents(sale_date);
CREATE INDEX idx_partsales_part_sale ON PartSales(part_id, sale_id);

-- Trigger: Set part status to 'sold' when inserted in PartSales
DELIMITER $$
CREATE TRIGGER trg_set_part_status_sold
AFTER INSERT ON PartSales
FOR EACH ROW
BEGIN
  UPDATE Parts SET status = 'sold'
  WHERE part_id = NEW.part_id;
```

```
END;
$$


-- Trigger: Reset part status to 'in stock' only if no sales remain for the part after deletion from PartSales
CREATE TRIGGER trg_reset_part_status
AFTER DELETE ON PartSales
FOR EACH ROW
BEGIN
  IF NOT EXISTS (SELECT 1 FROM PartSales WHERE part_id = OLD.part_id) THEN
    UPDATE Parts SET status = 'in stock' WHERE part_id = OLD.part_id;
  END IF;
END;
$$
DELIMITER ;
```