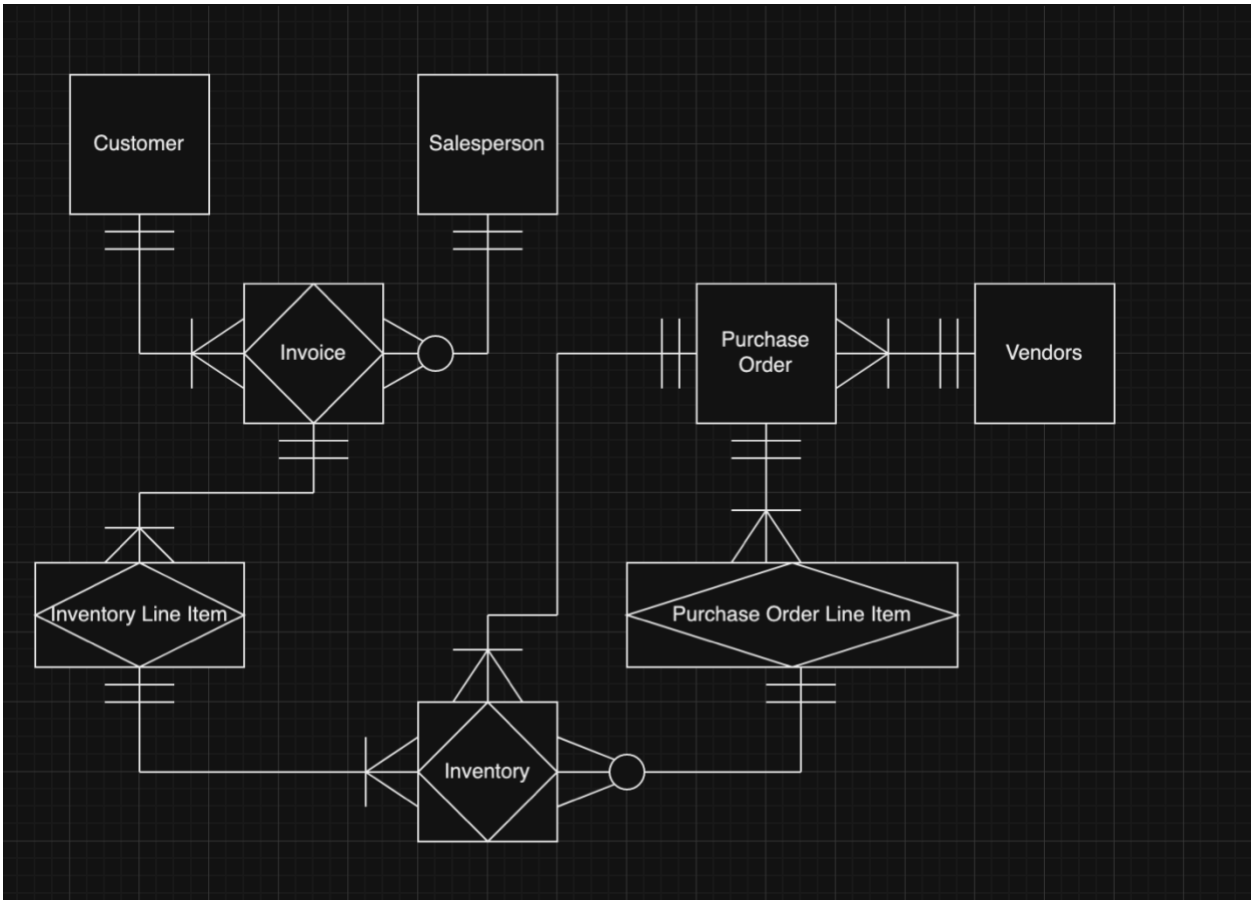
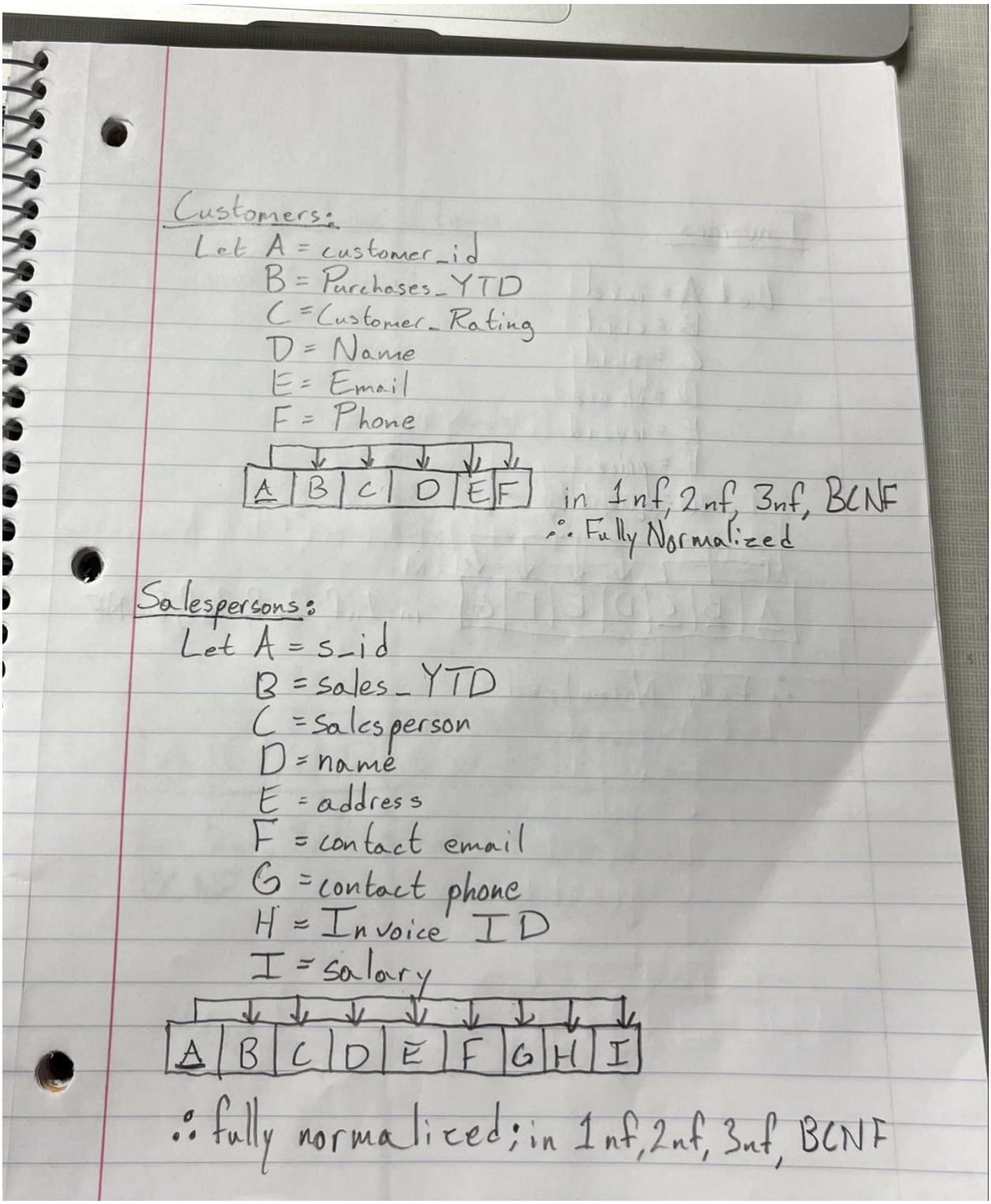
DS3860-003
Trevor Rowland, Matt Mills, Kenton Crockett
Final Project

ACME Inc. Database

Entity Relationship Diagram:

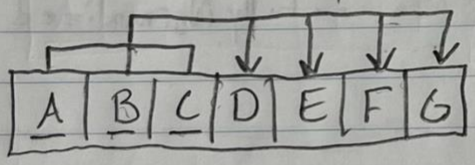


Normalization Tables:



Invoices

Let A = i_id
B = c_id
C = s_id
D = ili_id
E = inv_id
F = price
G = quantity

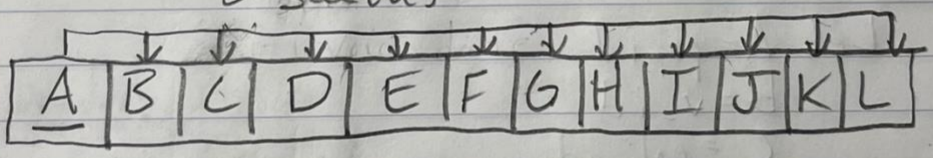


in 1nf, 2nf, 3nf, BCNF

∴ Fully Normalized

Inventory Line Items

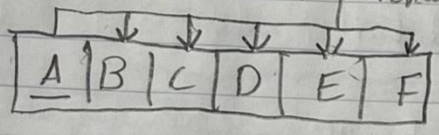
Let A = ili_id
B = name
C = quantity
D = price
E = product code
F = notes
G = shipping details
H = currency
I = payment terms
J = date issued
K = date filled
L = status



1nf, 2nf, 3nf, ∴ Fully Normalized
BCNF

Inventory

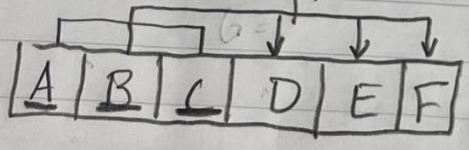
Let A = inv_id
B = ili_id
C = poli_id
D = quantity
E = total sales YTD
F = total purchases YTD



1nf, 2nf, 3nf, ∴ Fully Normalized
BCNF

Purchase Orders

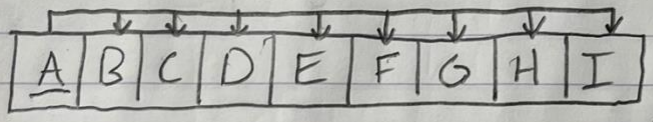
Let A = po_id
B = v_id
C = poli_id
D = inv_id
E = quantity
F = price



1nf, 2nf, 3nf, BCNF
∴ Fully Normalized

Purchase Order Line Items

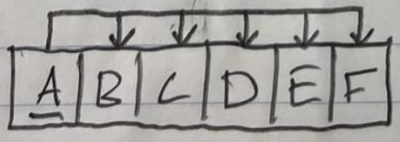
Let A = poli_id
B = name
C = quantity
D = price
E = date_issued
F = delivery date
G = payment method
H = shipping address
I = order status



in 1nf, 2nf, 3nf, BCNF
∴ Fully Normalized

Vendors

Let A = v_id
B = name
C = purchases_YTD
D = address
E = contact email
F = contact phone



in 1nf, 2nf, 3nf, BCNF
∴ Fully Normalized

SQL Code:

```
-- SCHEMA 4: ACME INC

--------------------------------------------------------------------------------------------
drop table Customers cascade constraints;
drop table Salespersons cascade constraints;
drop table Invoices cascade constraints;
drop table Inventory_Line_Items cascade constraints;
drop table Purchase_Orders cascade constraints;
drop table Vendors cascade constraints;
drop table Purchase_Order_Line_Items cascade constraints;
drop table Inventory cascade constraints;

BEGIN
  DBMS_OUTPUT.PUT_LINE('Tables Dropped.');
  DBMS_OUTPUT.PUT_LINE('Creating Tables...');
END;
/

--------------------------------------------------------------------------------------------
-- Create Customer, Salesperson and Invoice Tables
create table Customers(
  c_id char(10) NOT NULL PRIMARY KEY,
  purchases_YTD decimal(10,2),
  customer_rating int,
  name char(10),
  location char(50),
  contact_email char(30),
  contact_phone char(15)
);

create table Salespersons(
  s_id char(10) NOT NULL PRIMARY KEY,
  sales_YTD decimal(10,2),
  sales_performance int,
  name char(50),
  address char(20),
  contact_email char(30),
  contact_phone char(15),
  Invoice_ID char(10),
  salary decimal(10,2)
);
```

```
create table Invoices(
  i_id char(10) NOT NULL,
  c_id char(10) NOT NULL,
  s_id char(10) NOT NULL,
  ili_id char(10) NOT NULL,
  inv_id char(10) NOT NULL,
  price decimal(10,2),
  quantity int,
  PRIMARY KEY (i_id, c_id, s_id)
);

create table Inventory_Line_Items(
    ili_id char(10) NOT NULL PRIMARY KEY,
    name char(20),
    quantity int,
    price decimal(10,2),
    product_code char(10),
    notes_comments char(20),
    shipping_details char(20),
    currency char(15),
    payment_terms char(25),
    date_issued char(25),
    date_filled char(25),
    status char(20),
    invoice_date char(11)
);

create table Inventory(
            inv_id char(10) NOT NULL PRIMARY KEY,
    ili_id char(10) NOT NULL,
    poli_id char(10) NOT NULL,
    quantity int,
    total_sales_YTD decimal(10,2),
    total_purchases_YTD decimal(10,2)
);

BEGIN
  DBMS_OUTPUT.PUT_LINE('Added Customer, Salesperson, ILI, Inventory and Invoice Tables');
  DBMS_OUTPUT.PUT_LINE('Adding Foreign Keys for these tables...');
END;
/

-----------------------------------------------------------------------------------
-- Link Customers and Salespersons Tables to Invoice
```

```
alter table Invoices
   add constraint fk_customer_order
         foreign key (c_id) references Customers(c_id);

alter table Invoices
         add constraint fk_salesperson_order
         foreign key (s_id) references Salespersons(s_id);

alter table Invoices
         add constraint fk_ili_invoice
         foreign key (ili_id) references Inventory_Line_Items(ili_id);

alter table Invoices
   add constraint fk_quantity_after_purchase
   foreign key (inv_id) references Inventory(inv_id);

-- Connect Inventory to Inventory Line Items
alter table Inventory
   add constraint fk_ili
         foreign key (ili_id) references Inventory_Line_Items(ili_id);

BEGIN
  DBMS_OUTPUT.PUT_LINE('Foreign Keys Added.');
  DBMS_OUTPUT.PUT_LINE('Adding Invoice Triggers...');
END;
/

-- Update Purchases_YTD when a new Invoice is generated
create or replace trigger update_purchases_ytd_trigger
after insert on Invoices
for each row
begin
  update Customers
  set purchases_YTD = purchases_YTD + (:NEW.quantity * :NEW.price)
  where c_id = :NEW.c_id;
end;
/

create or replace trigger update_customer_rewards
after insert on Invoices
for each row
begin
  update Customers
  set customer_rating = customer_rating + 1
```

```
  where c_id = :NEW.c_id;
end;
/

-- Update Sales_YTD when a new Invoice is generated
CREATE OR REPLACE TRIGGER update_sales_ytd_trigger
AFTER INSERT ON Invoices
FOR EACH ROW
BEGIN
  UPDATE Salespersons
  SET sales_YTD = sales_YTD + (:NEW.quantity * :NEW.price)
  WHERE s_id = :NEW.s_id;
END;
/

CREATE OR REPLACE TRIGGER update_sales_performance_trigger
AFTER INSERT ON Invoices
FOR EACH ROW
BEGIN
  UPDATE Salespersons
  SET sales_performance = sales_performance + 1
  WHERE s_id = :NEW.s_id;
END;
/

CREATE OR REPLACE TRIGGER update_sales_ytd_trigger
AFTER INSERT ON Invoices
FOR EACH ROW
BEGIN
  UPDATE Inventory
  SET quantity = quantity - :NEW.quantity
  WHERE inv_id = :NEW.inv_id;
END;
/

BEGIN
  DBMS_OUTPUT.PUT_LINE('Triggers Added.');
END;
/
---------------------------------------------------------------------------------------------

-- Purchase Order Section:
BEGIN
  DBMS_OUTPUT.PUT_LINE('Adding Purchase Order Handling...');
```

```
END;
/
create table Purchase_Orders(
        po_id char(10) NOT NULL,
        v_id char(10) NOT NULL,
        poli_id char(10) NOT NULL,
  inv_id char(10) NOT NULL,
        quantity int,
        price decimal(10,2),
  PRIMARY KEY (po_id, v_id, poli_id)
);

create table Vendors(
  v_id char(10) NOT NULL PRIMARY KEY,
  name char(25),
  purchases_YTD int,
  address char(20),
  contact_email char(30),
  contact_phone char(15),
);

create table Purchase_Order_Line_Items(
  poli_id char(10) NOT NULL PRIMARY KEY,
  name char(20),
  quantity int,
  price decimal(10,2), -- price per unit
  date_issued char(25),
  delivery_date char(10),
  payment_method char(15),
  shipping_address char(20),
  order_status char(15)
);


BEGIN
  DBMS_OUTPUT.PUT_LINE('Purchase Order, Product, POLI and Vendors Added.');
  DBMS_OUTPUT.PUT_LINE('Adding Foreign Keys...');
END;
/
-- Connect Purchase Orders to POLIs and Vendors
alter table Purchase_Orders
        add constraint fk_vendor_order
        foreign key (v_id) references Vendors(v_id);
```

```
alter table Purchase_Orders
	add constraint fk_poli
	foreign key (poli_id) references Purchase_Order_Line_Items(poli_id);

alter table Purchase_Orders
   add constraint fk_inv_to_po
   foreign key (inv_id) references Inventory(inv_id);

BEGIN
  DBMS_OUTPUT.PUT_LINE('Foreign Keys Added.');
  DBMS_OUTPUT.PUT_LINE('Adding Purchase Order Triggers...');
END;
/
-- Triggers when a new Purchase Order is created
create or replace trigger update_poli_from_po
after insert on Purchase_Orders
for each row
begin
	update Purchase_Order_Line_Items
		set quantity = quantity + (:NEW.quantity)
		where poli_id = :NEW.poli_id;
end;
/

create or replace trigger update_inventory_from_po
after insert on Purchase_Orders
for each row
begin
	update Inventory
		set quantity = quantity + (:NEW.quantity)
		where inv_id = :NEW.inv_id;
end;
/

BEGIN
  DBMS_OUTPUT.PUT_LINE('Purchase Order Triggers Finished.');
  DBMS_OUTPUT.PUT_LINE('Add Testing Below...');
END;
/
```