Course Submission Cover Sheet Module:
CC5051 Databases
Assignment no: 001
Weighting: 50% of the module mark
Deadline: 11th (Wednesday) of January 2023, 23:59 pm
Module Leader: Dr Qicheng Yu Student ID: 20044497

LONDON METROPOLITAN UNIVERSITY

Introduction

In this coursework, we will use SQL Developer and Oracle Database to design and build a database for a retail invoice. We will start by creating a table to store information about the items purchased, including the item name, quantity, and price. We will then create another table to store the invoice details, such as the customer name, invoice number, and total amount due.

Once the tables are created, we will populate them with sample data and use SQL queries to extract information from the invoice. For example, we will write a query to retrieve the list of items and their quantities and prices, as well as the total number of items and the total cost of the purchase. We will also learn how to use SQL statements to insert, update, and delete data from the tables.

By the end of this coursework, you will have gained practical experience in using SQL Developer and Oracle Database to design and build a database for a retail invoice, and in using SQL queries to manipulate and extract information from the database.

Section 1.1:

Diagrams

After studying the Mobile for You case study I have identified the following entities and I have drawn an ER diagram to display the relationship between these entities accordingly.

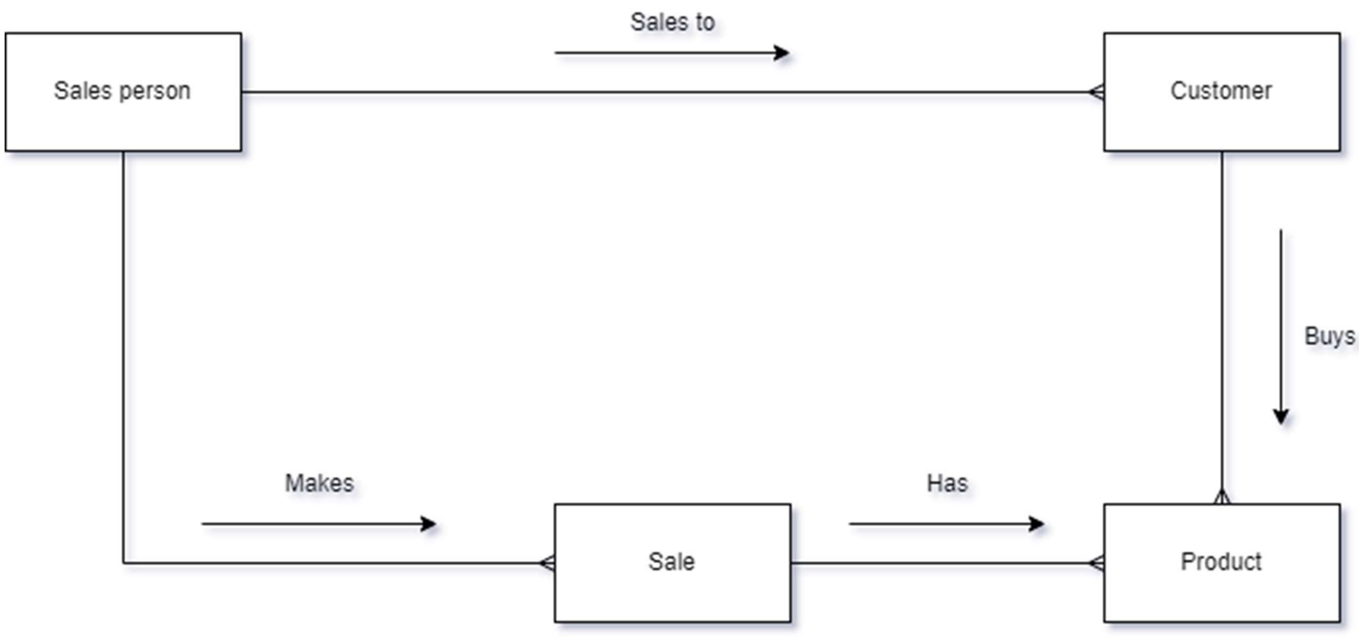


Figure 1: Basic conceptual diagram

During the identification process of the data required to build the conceptual ER diagram, a series of assumptions have been made.

Assumption a.

We assume that the salesperson sells to one or more customers.

Assumption b.

We assume that the salesperson makes one or multiple sales.

Assumption c.

We assume that a sale contains one or more products.

Assumption d.

We assume that a customer can buy one or multiple products.

Section 1.2:

After further inspection of the case study sale receipt and text, further data was extracted, and a more detailed ER diagram was drawn containing the entity attributes.

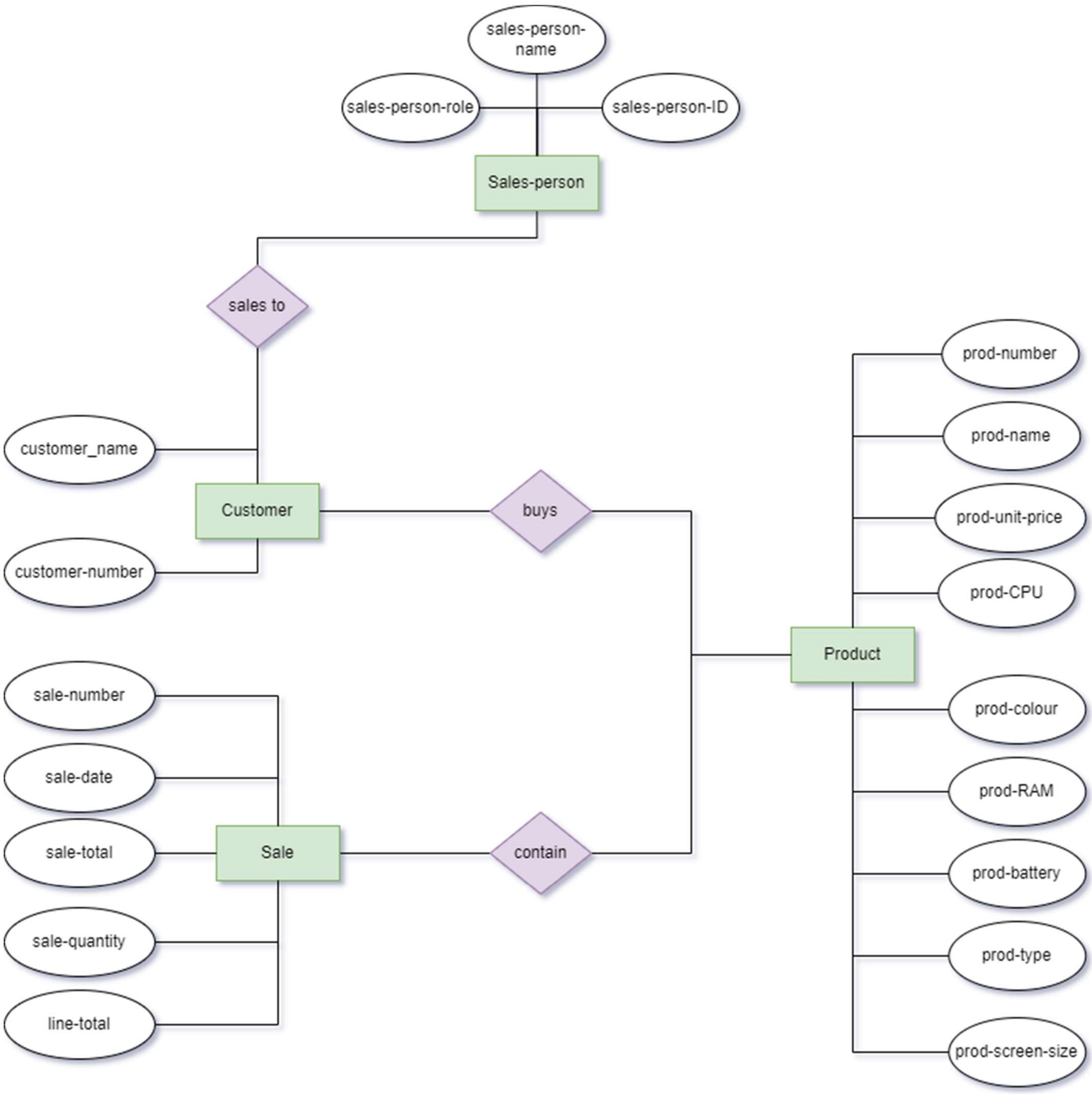


Figure 2: Detailed conceptual diagram

Section 1.3:

Taking into account the case study given to us through the coursework I have identified the following data in an unnormalized form.

UNF

Sale (sale-no, sale-date, cust-no, cust-name, sales-id, sales-person, {prod-no, prod-name, prod-type, prod-unit-price, prod-OS, prod-screen-size, prod-CPU, prod-RAM, prod-battery, prod-colour, order-quantity, line-total}, total-price)

To make sense of the data I will have to pass the data through the normalization steps. In consequence, to put the data in the first normal form, I must remove the repeating groups of data.

1 NF

Sale (sale-no, sale-date, cust-no, cust-name, sales-id, sales-name, total-price)

Sale-line (sale-no*, prod-no, prod-name, prod-type, prod-unit-price, prod-OS, prod-screen-size, prod-CPU, prod-RAM, prod-battery, prod-colour, order-quantity, line-total)

For the first normal form we end up with 2 entities with their attributes, the first entity has one primary key and the second entity has one primary key and one foreign key

For the second normal form, we need to remove all partial dependencies as follows.

2 NF

Sale (sale-no, sale-date, cust-no, cust-name, sales-id, sales-person, sales-person-role, line-total)
Product (prod-no, prod-name, prod-type, prod-unit-price, prod-OS, prod-screen-size, prod-CPU, prod-RAM, prod-battery, prod-colour)
Order (sale-no*, prod-no*, order-quantity, total-price)

Going through the second normal form normalization process we end up with 3 entities first entity Sale has one primary key, the second entity has one primary key and the third entity has 2 foreign keys.

For the third normal form, we need to remove all transitive dependencies as follows.

3 NF

Sale (sale-no, sale-date, staff-id*, cust-no*, line-total)
Staff (staff-id, staff-person, staff-role)
Customer (cust-no, cust-name)
Product (prod-no, prod-name, prod-type, prod-unit-price, prod-OS, prod-screen-size, prod-CPU, prod-RAM, prod-battery, prod-colour)
Order (sale-no*, prod-no*, order-quantity, total-price)

Going through the third normal form we end up with five distinct entities that each have distinct primary keys and some of them have foreign keys to relate to the other entities.

After passing the data through the normalization process and identifying the distinct entities we now must draw an ER (Entity Relationship) Diagram showing the relationship between these entities and all their attributes.

During the normalization steps, additional assumptions have been made to organize the data and reduce redundancy.

Assumption e.

Since the roles of the people working in the shop are different, it would be more appropriate to rename the initially identified entity **Sales-person** to the more general term **Staff** and rename all properties accordingly.

Section 1.4:

In this section, another form of the normalization process is shown, mirroring the process in the previous section.

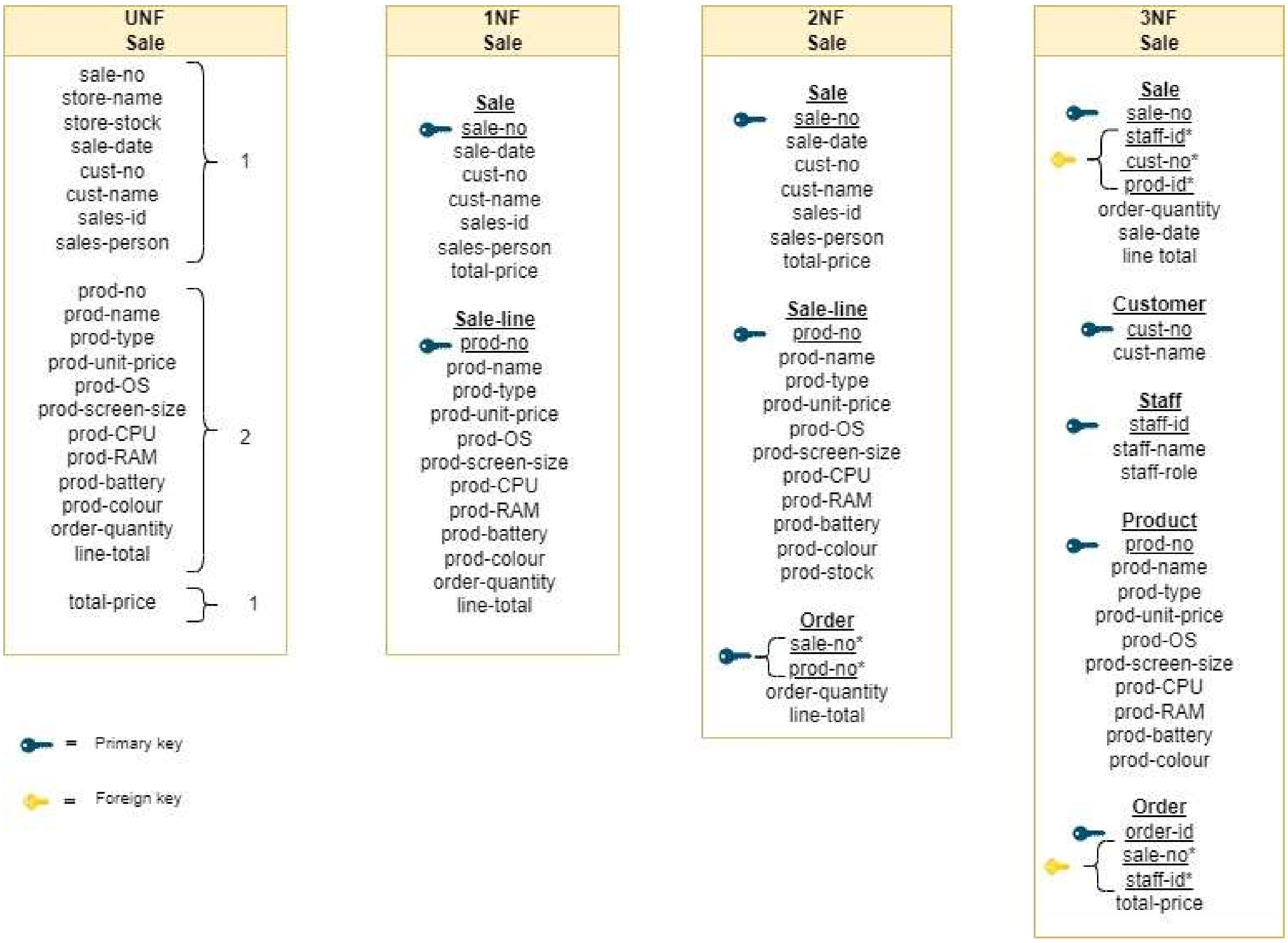
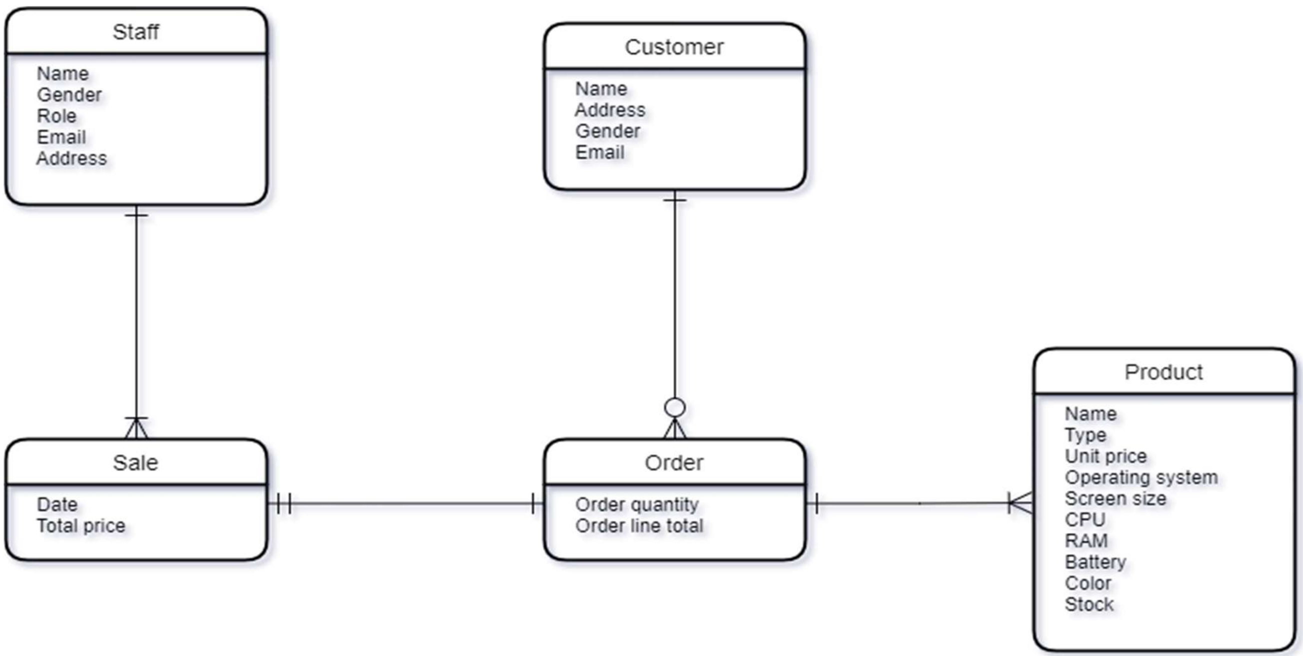


Figure 3: Normalization process in table form

Section 1.5:

To better understand the process of designing the database for the project a Conceptual data model has been drawn to enrich the conceptual model by defining explicit columns in each entity and by introducing transactional entities the relationship between the entities becomes clearer.



Section 1.6:

Figure 4: Logical ER Diagram

The Physical data model represents the actual design of the relational database that we need to build. The physical data model aims to elaborate the logical data model by assigning each column with type, length, nullable and many more. Because an Entity relationship diagram represents how data should be structured and related it is essential to consider the conventions and restrictions of actual database systems.

During the design process, additional entities have been identified as being necessary to the model to make better sense of the data. In consequence, other assumptions have been made and the entities have been added to the design process.

Assumption f.

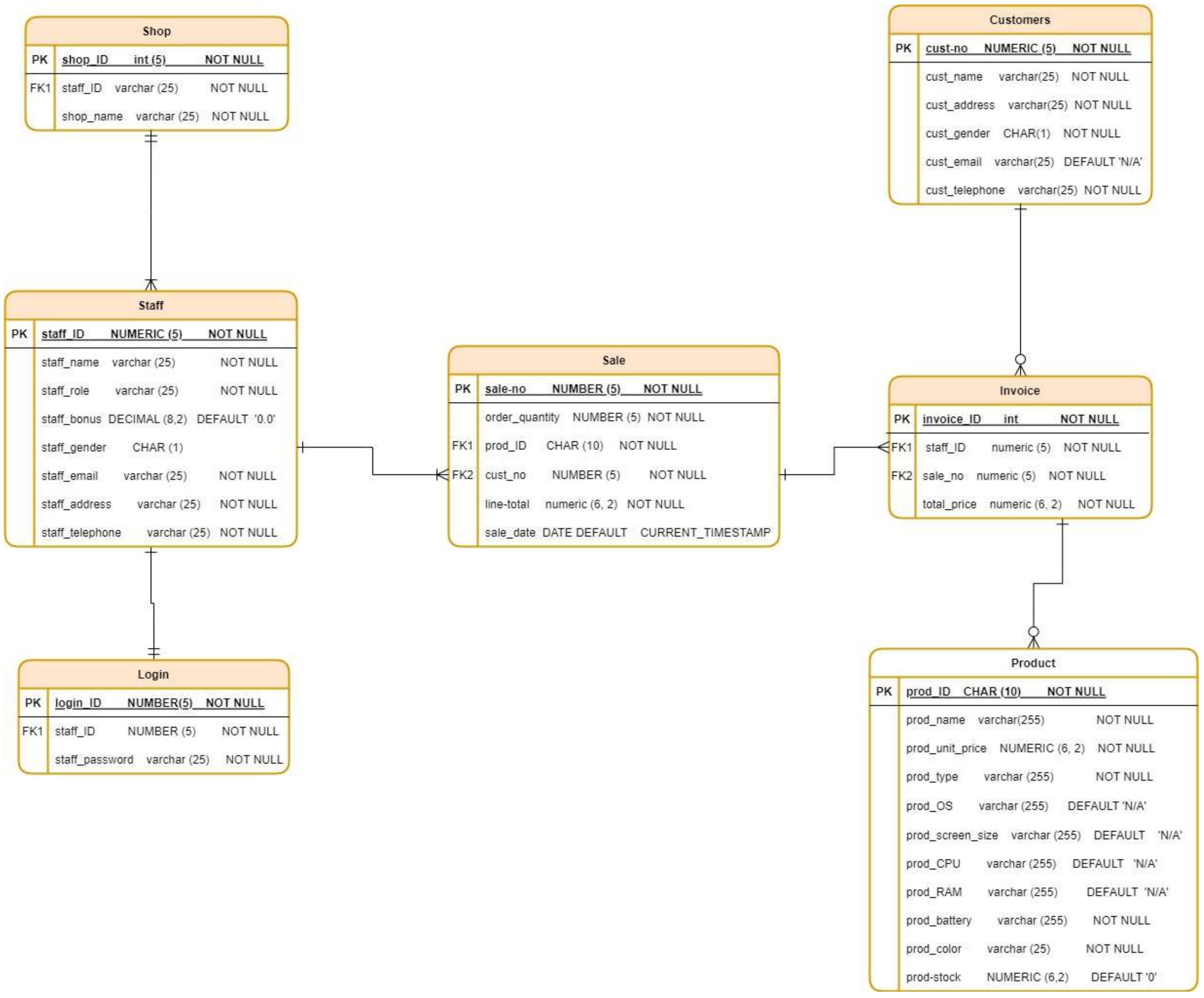
Knowing that the Mobile Store is a physical store to be able to make sales the staff members will have to log in to a system that will allow them access and track information regarding stock availability and other information relating to the Mobile Store. The resulting entity of the made assumption is **Login** (login-id, staff-id*, login-password).

Assumption g.

An extra assumption is that because the given case study mentions a physical store we can also assume that the store has a name and a store id that makes it unique on the high street and this would also be a way of keeping track of the staff members. The resulting entity is named **Store** (store-id, store-name, staff-id*, staff-state)

Section 1.7:

As seen in this section the final ER diagram has been drawn and extra assumptions have been made to model the retail store accurately.


Shop
PK shop_ID int (5) NOT NULL
FK1 staff_ID varchar (25) NOT NULL
shop_name varchar (25) NOT NULL
Customers
PK cust-no NUMERIC (5) NOT NULL
cust_name varchar(25) NOT NULL
cust_address varchar(25) NOT NULL
cust_gender CHAR(1) NOT NULL
cust_email varchar(25) DEFAULT 'N/A'
cust_telephone varchar(25) NOT NULL
Staff
PK staff_ID NUMERIC (5) NOT NULL
staff_name varchar (25) NOT NULL
staff_role varchar (25) NOT NULL
staff_bonus DECIMAL (8,2) DEFAULT '0.0'
staff_gender CHAR (1)
staff_email varchar (25) NOT NULL
staff_address varchar (25) NOT NULL
staff_telephone varchar (25) NOT NULL
Sale
PK sale-no NUMBER (5) NOT NULL
order_quantity NUMBER (5) NOT NULL
FK1 prod_ID CHAR (10) NOT NULL
FK2 cust_no NUMBER (5) NOT NULL
line-total numeric (6, 2) NOT NULL
sale_date DATE DEFAULT CURRENT_TIMESTAMP
Invoice
PK invoice_ID int NOT NULL
FK1 staff_ID numeric (5) NOT NULL
FK2 sale_no numeric (5) NOT NULL
total_price numeric (6, 2) NOT NULL
Login
PK login_ID NUMBER(5) NOT NULL
FK1 staff_ID NUMBER (5) NOT NULL
staff_password varchar (25) NOT NULL
Product
PK prod_ID CHAR (10) NOT NULL
prod_name varchar(255) NOT NULL
prod_unit_price NUMERIC (6, 2) NOT NULL
prod_type varchar (255) NOT NULL
prod_OS varchar (255) DEFAULT 'N/A'
prod_screen_size varchar (255) DEFAULT 'N/A'
prod_CPU varchar (255) DEFAULT 'N/A'
prod_RAM varchar (255) DEFAULT 'N/A'
prod_battery varchar (255) NOT NULL
prod_color varchar (25) NOT NULL
prod-stock NUMERIC (6,2) DEFAULT '0'

Ass

Figure 5: Final physical ER diagram

As a final assumption, we assume that the database is designed to store additional information for some of the entities for record keeping. Information like gender, email, address and telephone number for both staff members and customers have been added.

Assumption i.

In order to make sense of the data and to make it more readable I have decided to rename the **Order** table in to **Invoice**

Table creation

```
-- Drop table if it exists
DROP TABLE Shop cascade constraints;
DROP TABLE Staff cascade constraints;
DROP TABLE Login cascade constraints;
DROP TABLE Customer cascade constraints;
DROP TABLE Sale cascade constraints;
DROP TABLE Invoice cascade constraints;
DROP TABLE Product cascade constraints;




-- Create the Staff table
CREATE TABLE Staff (
  staff_ID NUMERIC (5) NOT NULL,
  staff_name VARCHAR(255) NOT NULL,
  staff_role VARCHAR(255) NOT NULL,
  staff_bonus DECIMAL(8,2) DEFAULT '0.0',
  staff_gender CHAR(1),
  staff_email VARCHAR(255),
  staff_address VARCHAR2(255),
  staff_telephone VARCHAR(20),
  CONSTRAINT PK_Staff_ID PRIMARY KEY (staff_ID)
);

-- Insert data into the Staff table
INSERT INTO Staff (staff_ID, staff_name, staff_role, staff_bonus, staff_gender, staff_email, staff_address,
staff_telephone)
VALUES (10000, 'John Smith', 'Manager', 1000, 'M', 'johnsmith@gmail.com', '1234 Main St', '123-456-
7890');
INSERT INTO Staff (staff_ID, staff_name, staff_role, staff_gender, staff_email, staff_address,
staff_telephone)
VALUES (10100, 'Jane Doe', 'Employee', 'F', 'janedoe@gmail.com', '5678 Main St', '098-765-4321');
INSERT INTO Staff (staff_ID, staff_name, staff_role, staff_gender, staff_email, staff_address,
staff_telephone)
VALUES (10200, 'Bob Johnson', 'Employee', 'M', 'bobjohnson@gmail.com', '1011 Main St', '111-222-3333');
INSERT INTO Staff (staff_ID, staff_name, staff_role, staff_gender, staff_email, staff_address,
staff_telephone)
VALUES (10300, 'Amy Smith', 'Employee', 'F', 'amysmith@gmail.com', '1212 Main St', '444-555-6666');
INSERT INTO Staff (staff_ID, staff_name, staff_role, staff_gender, staff_email, staff_address,
staff_telephone)
VALUES (10400, 'Mike Jones', 'Employee', 'M', 'mikejones@gmail.com', '1313 Main St', '777-888-9999');


--Create shop table
CREATE TABLE Shop(
    shop_ID NUMBER (6) NOT NULL,
    staff_ID NUMBER (5) NOT NULL,
    shop_name varchar2(255) NOT NULL,
    CONSTRAINT PK_shop  PRIMARY KEY (shop_ID),
    CONSTRAINT Staff_ID_FK FOREIGN KEY (staff_ID) REFERENCES Staff (staff_ID)
);

INSERT INTO Shop (shop_ID, staff_ID, shop_name)
```

```
VALUES (1, 10000, 'Gadget Shop');


CREATE TABLE Login(
    login_ID NUMBER (5) NOT NULL,
    staff_ID NUMBER (5) NOT NULL,
    staff_password varchar2(255),
    CONSTRAINT PK_login_ID PRIMARY KEY (login_ID, staff_ID),
    CONSTRAINT FK_Stafff_ID FOREIGN KEY (staff_ID) REFERENCES Staff(staff_ID)
);

INSERT INTO Login (login_ID, staff_ID, staff_password)
VALUES (1, 10000, 'P@ssw0rd1');
INSERT INTO Login (login_ID, staff_ID, staff_password)
VALUES (2, 10100, 'D0eJ@n3#');
INSERT INTO Login (login_ID, staff_ID, staff_password)
VALUES (3, 10200, 'B0bJ0hn$0n');
INSERT INTO Login (login_ID, staff_ID, staff_password)
VALUES (4, 10300, 'AmY$m1th');
INSERT INTO Login (login_ID, staff_ID, staff_password)
VALUES (5, 10400, 'M!k3J0n3s');


CREATE TABLE Customer (
  cust_no NUMBER (5) NOT NULL,
  cust_name varchar2(255) NOT NULL,
  cust_address varchar2(255),
  cust_gender CHAR(1),
  cust_email varchar2(255) DEFAULT 'N/A',
  cust_telno varchar2(20),
  CONSTRAINT PK_cust_no PRIMARY KEY (cust_no)
);

INSERT INTO Customer (cust_no, cust_name, cust_address, cust_gender, cust_email, cust_telno)
VALUES (1, 'John Doe', '123 Main St', 'M', 'johndoe@gmail.com', '555-555-5555');
INSERT INTO Customer (cust_no, cust_name, cust_address, cust_gender, cust_email, cust_telno)
VALUES (2, 'Jane Smith', '456 Park Ave', 'F', 'janesmith@gmail.com', '555-555-5556');
INSERT INTO Customer (cust_no, cust_name, cust_address, cust_gender, cust_email, cust_telno)
VALUES (3, 'Bob Johnson', '789 Elm St', 'M', 'bobjohnson@gmail.com', '555-555-5557');
INSERT INTO Customer (cust_no, cust_name, cust_address, cust_gender, cust_email, cust_telno)
VALUES (4, 'Amy Williams', '321 Oak St', 'F', 'amywilliams@gmail.com', '555-555-5558');
INSERT INTO Customer (cust_no, cust_name, cust_address, cust_gender, cust_email, cust_telno)
VALUES (5, 'Mike Brown', '147 Pine St', 'M', 'mikebrown@gmail.com', '555-555-5559');
INSERT INTO Customer (cust_no, cust_name, cust_address, cust_gender, cust_email, cust_telno)
VALUES (6, 'Emily Davis', '258 Cedar St', 'F', 'emilydavis@gmail.com', '555-555-5560');
INSERT INTO Customer (cust_no, cust_name, cust_address, cust_gender, cust_email, cust_telno)
VALUES (7, 'Jacob Miller', '369 Birch St', 'M', 'jacobmiller@gmail.com', '555-555-5561');
INSERT INTO Customer (cust_no, cust_name, cust_address, cust_gender, cust_email, cust_telno)
VALUES (8, 'Sophia Garcia', '159 Maple St', 'F', 'sophiagarcia@gmail.com', '555-555-5562');
INSERT INTO Customer (cust_no, cust_name, cust_address, cust_gender, cust_email, cust_telno)
VALUES (9, 'Michael Rodriguez', '753 Cherry St', 'M', 'michaelrodriguez@gmail.com', '555-555-5563');
INSERT INTO Customer (cust_no, cust_name, cust_address, cust_gender, cust_email, cust_telno)
VALUES (10, 'Madison Thompson', '951 Apple St', 'F', 'madisonthompson@gmail.com', '555-555-5564');


CREATE TABLE Product(
    prod_ID CHAR (10) NOT NULL,
    prod_name VARCHAR2(255) NOT NULL,
    prod_type VARCHAR2(255) NOT NULL,
    prod_unit_price NUMERIC(6,2) NOT NULL,
    prod_OS VARCHAR2 (255) DEFAULT 'N/A',
    prod_screen_size VARCHAR2(255) DEFAULT 'N/A',
    prod_CPU VARCHAR2(25) DEFAULT 'N/A',
    prod_RAM VARCHAR2(25) DEFAULT 'N/A',
    prod_battery VARCHAR2(25) NOT NULL,
    prod_color VARCHAR2(25) NOT NULL,
    prod_stock NUMERIC(6,2)DEFAULT '0',
    CONSTRAINT PK_prod_ID PRIMARY KEY (prod_ID)
);

INSERT INTO Product (prod_ID, prod_name, prod_type, prod_unit_price, prod_OS, prod_screen_size,
prod_CPU, prod_RAM, prod_battery, prod_color, prod_stock)
VALUES ('S21G', 'Samsung Galaxy S21', 'Smartphone', 799.99, 'Android', '6.2 inches', 'Snapdragon 888',
'8GB', '4000mAh', 'Phantom Gray', 25);
INSERT INTO Product (prod_ID, prod_name, prod_type, prod_unit_price, prod_OS, prod_screen_size,
prod_CPU, prod_RAM, prod_battery, prod_color, prod_stock)
VALUES ('S21P', 'Samsung Galaxy S21 Plus', 'Smartphone', 999.99, 'Android', '6.7 inches', 'Snapdragon
888', '8GB', '4800mAh', 'Phantom Black', 15);
INSERT INTO Product (prod_ID, prod_name, prod_type, prod_unit_price, prod_OS, prod_screen_size,
prod_CPU, prod_RAM, prod_battery, prod_color, prod_stock)
VALUES ('S21U', 'Samsung Galaxy S21 Ultra', 'Smartphone', 1199.99, 'Android', '6.8 inches', 'Snapdragon
888', '12GB', '5000mAh', 'Phantom Black', 10);
INSERT INTO Product (prod_ID, prod_name, prod_type, prod_unit_price, prod_OS, prod_screen_size,
prod_CPU, prod_RAM, prod_battery, prod_color, prod_stock)
VALUES ('IP12', 'iPhone 12', 'Smartphone', 799.99, 'iOS', '6.1 inches', 'A14 Bionic', '4GB', '2815mAh',
'Graphite', 30);
INSERT INTO Product (prod_ID, prod_name, prod_type, prod_unit_price, prod_OS, prod_screen_size,
prod_CPU, prod_RAM, prod_battery, prod_color, prod_stock)
VALUES ('IP12P', 'iPhone 12 Pro', 'Smartphone', 999.99, 'iOS', '6.1 inches', 'A14 Bionic', '6GB', '2815mAh',
'Gold', 20);
INSERT INTO Product (prod_ID, prod_name, prod_type, prod_unit_price, prod_OS, prod_screen_size,
prod_CPU, prod_RAM, prod_battery, prod_color, prod_stock)
VALUES ('IP12PM', 'iPhone 12 Pro Max', 'Smartphone', 1099.99, 'iOS', '6.7 inches', 'A14 Bionic', '6GB',
'3375mAh', 'Pacific Blue', 25);
INSERT INTO Product (prod_ID, prod_name, prod_type, prod_unit_price, prod_OS, prod_screen_size,
prod_CPU, prod_RAM, prod_battery, prod_color, prod_stock)
VALUES ('SU128', 'Sandisk Ultra 128GB USB 3.0 Flash Drive', 'External Storage', 19.99, 'N/A', 'N/A', 'N/A',
'N/A', 'N/A', 'N/A', 100);
INSERT INTO Product (prod_ID, prod_name, prod_type, prod_unit_price, prod_OS, prod_screen_size,
prod_CPU, prod_RAM, prod_battery, prod_color, prod_stock)
VALUES ('SE2TB', 'Seagate Expansion 2TB Portable External Hard Drive', 'External Storage', 69.99, 'N/A',
'N/A', 'N/A', 'N/A', 'N/A', 'N/A', 80);
INSERT INTO Product (prod_ID, prod_name, prod_type, prod_unit_price, prod_OS, prod_screen_size,
prod_CPU, prod_RAM, prod_battery, prod_color, prod_stock)
VALUES ('AN10000', 'Anker PowerCore 10000mAh Portable Charger', 'Powerbank', 29.99, 'N/A', 'N/A',
'N/A', 'N/A', 'N/A', 'N/A', 50);
INSERT INTO Product (prod_ID, prod_name, prod_type, prod_unit_price, prod_OS, prod_screen_size,
prod_CPU, prod_RAM, prod_battery, prod_color, prod_stock)
VALUES ('AUCLBL', 'Aukey USB-C to Lightning Cable', 'Charging Cable', 14.99, 'N/A', 'N/A', 'N/A', 'N/A',
'N/A', 'N/A', 75);
INSERT INTO Product (prod_ID, prod_name, prod_type, prod_unit_price, prod_OS, prod_screen_size,
prod_CPU, prod_RAM, prod_battery, prod_color, prod_stock)
VALUES ('SPIGI12', 'Spigen Tough Armor iPhone 12 Case', 'Phone Case', 14.99, 'N/A', 'N/A', 'N/A', 'N/A',
'N/A', 'N/A', 100);
INSERT INTO Product (prod_ID, prod_name, prod_type, prod_unit_price, prod_OS, prod_screen_size,
prod_CPU, prod_RAM, prod_battery, prod_color, prod_stock)
VALUES ('RIFUS21', 'Ringke Fusion-X Samsung Galaxy S21 Case', 'Phone Case', 9.99, 'N/A', 'N/A', 'N/A',
'N/A', 'N/A', 'N/A', 120);
```

```
CREATE TABLE Sale(
    sale_no NUMBER (5) NOT NULL,
    order_quantity INTEGER NOT NULL,
    staff_ID NUMERIC (5) NOT NULL,
    cust_no NUMBER (5) NOT NULL,
    prod_ID CHAR (10) NOT NULL,
    line_total_price NUMERIC(10, 2),
    sale_date DATE DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT PK_sale_no PRIMARY KEY (sale_no),
    CONSTRAINT FK_cust_no FOREIGN KEY (cust_no) REFERENCES Customer (cust_no),
    CONSTRAINT FK_prod_id FOREIGN KEY (prod_ID) REFERENCES Product (prod_ID),
    CONSTRAINT FK_staffffff_id FOREIGN KEY (staff_ID) REFERENCES Staff (staff_ID)
);

--First sale of products starts here
INSERT INTO Sale (staff_ID, sale_no, prod_ID, order_quantity, line_total_price, cust_no, sale_date)
VALUES (10100, 33100, 'S21G', 2, (SELECT prod_unit_price * 2 FROM Product WHERE prod_ID =
'S21G'),1 , DEFAULT);
INSERT INTO Sale (staff_ID, sale_no, prod_ID, order_quantity, line_total_price, cust_no, sale_date)
VALUES (10100, 33101, 'S21P', 3, (SELECT prod_unit_price * 3 FROM Product WHERE prod_ID =
'S21P'), 3, DEFAULT);
INSERT INTO Sale (staff_ID, sale_no, prod_ID, order_quantity, line_total_price, cust_no, sale_date)
VALUES (10200, 33102, 'RIFUS21', 1, (SELECT prod_unit_price * 1 FROM Product WHERE prod_ID =
'RIFUS21'), 3, DEFAULT);
--First sale of products ends here

--Second sale of products starts here
INSERT INTO Sale (staff_ID, sale_no, prod_ID, order_quantity, line_total_price, cust_no, sale_date)
VALUES (10300, 33103, 'AN10000', 1, (SELECT prod_unit_price * 1 FROM Product WHERE prod_ID =
'AN10000'), 2,  DEFAULT);
INSERT INTO Sale (staff_ID, sale_no, prod_ID, order_quantity, line_total_price, cust_no, sale_date)
VALUES (10400, 33104, 'IP12P', 3, (SELECT prod_unit_price * 3 FROM Product WHERE prod_ID =
'IP12P'), 10, DEFAULT);
INSERT INTO Sale (staff_ID, sale_no, prod_ID, order_quantity, line_total_price, cust_no, sale_date)
VALUES (10300, 33105, 'SE2TB', 2, (SELECT prod_unit_price * 2 FROM Product WHERE prod_ID =
'SE2TB'), 3, DEFAULT);
--second sale stops here

--third sale starts here
INSERT INTO Sale (staff_ID, sale_no, prod_ID, order_quantity, line_total_price, cust_no, sale_date)
VALUES (10100, 33106, 'AUCLBL', 4, (SELECT prod_unit_price * 4 FROM Product WHERE prod_ID =
'AUCLBL'), 1, DEFAULT);
INSERT INTO Sale (staff_ID, sale_no, prod_ID, order_quantity, line_total_price, cust_no, sale_date)
VALUES (10400, 33107, 'RIFUS21', 3, (SELECT prod_unit_price * 3 FROM Product WHERE prod_ID =
'RIFUS21'), 4, DEFAULT);
INSERT INTO Sale (staff_ID, sale_no, prod_ID, order_quantity, line_total_price, cust_no, sale_date)
VALUES (10300, 33108, 'IP12PM', 4, (SELECT prod_unit_price * 4 FROM Product WHERE prod_ID =
'IP12PM'), 5, DEFAULT);
INSERT INTO Sale (staff_ID, sale_no, prod_ID, order_quantity, line_total_price, cust_no, sale_date)
VALUES (10100, 33109, 'S21P', 2, (SELECT prod_unit_price * 2 FROM Product WHERE prod_ID =
'S21P'), 6, DEFAULT);
--third sale stops here

--fourth sale starts here
INSERT INTO Sale (staff_ID, sale_no, prod_ID, order_quantity, line_total_price, cust_no, sale_date)
VALUES (10300, 33110, 'IP12PM', 1, (SELECT prod_unit_price * 1 FROM Product WHERE prod_ID =
'IP12PM'), 7, DEFAULT);
INSERT INTO Sale (staff_ID, sale_no, prod_ID, order_quantity, line_total_price, cust_no, sale_date)
VALUES (10400, 33111, 'S21G', 5, (SELECT prod_unit_price * 5 FROM Product WHERE prod_ID =
'S21G'), 8, DEFAULT);
```

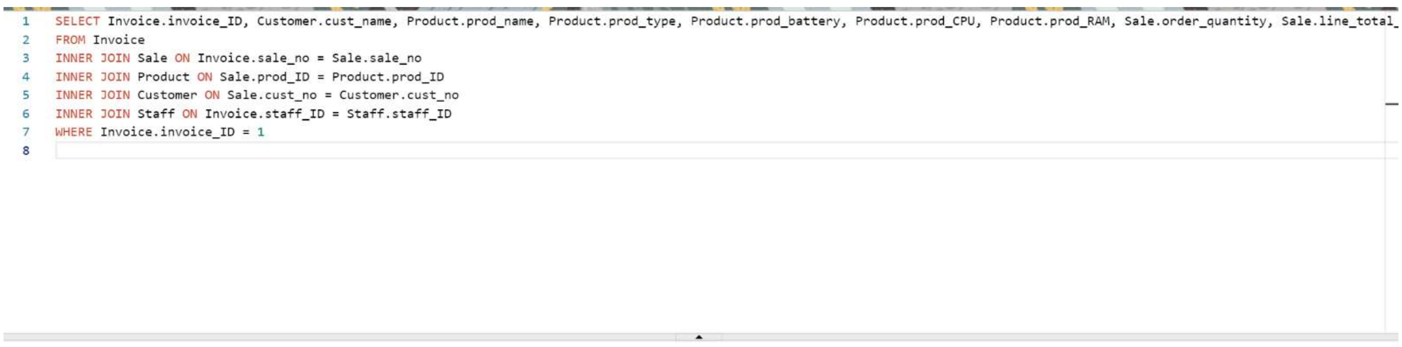
```
INSERT INTO Sale (staff_ID, sale_no, prod_ID, order_quantity, line_total_price, cust_no, sale_date)
VALUES (10400, 33112, 'IP12P', 2, (SELECT prod_unit_price * 2 FROM Product WHERE prod_ID =
'IP12P'), 2, DEFAULT);
INSERT INTO Sale (staff_ID, sale_no, prod_ID, order_quantity, line_total_price, cust_no, sale_date)
VALUES (10300, 33113, 'AN10000', 6, (SELECT prod_unit_price * 6 FROM Product WHERE prod_ID =
'AN10000'), 9, DEFAULT);
INSERT INTO Sale (staff_ID, sale_no, prod_ID, order_quantity, line_total_price, cust_no, sale_date)
VALUES (10100, 33114, 'S21U', 2, (SELECT prod_unit_price * 2 FROM Product WHERE prod_ID =
'S21U'), 8, DEFAULT);
--fourth sale ends here



CREATE TABLE Invoice(
   invoice_ID NUMBER (5) NOT NULL,
   staff_ID NUMERIC (5) NOT NULL,
   sale_no NUMBER (5) NOT NULL,
   total_price NUMERIC(10, 2) NOT NULL,
   CONSTRAINT PK_invoice_id PRIMARY KEY (invoice_ID),
   CONSTRAINT FK_staffff_ID FOREIGN KEY (staff_ID) REFERENCES Staff (staff_ID),
   CONSTRAINT FK_salee_no FOREIGN KEY (sale_no) REFERENCES Sale (sale_no)
);


INSERT INTO Invoice (invoice_ID, staff_ID, sale_no, total_price)
SELECT 1, S.staff_ID, SL.sale_no, SL.line_total_price
FROM Staff S, Sale SL
WHERE S.staff_ID = 10200 AND SL.sale_no = 33105;

INSERT INTO Invoice (invoice_ID, staff_ID, sale_no, total_price)
SELECT 2, S.staff_ID, SL.sale_no, SL.line_total_price
FROM Staff S, Sale SL
WHERE S.staff_ID = 10400 AND SL.sale_no = 33114;

INSERT INTO Invoice (invoice_ID, staff_ID, sale_no, total_price)
SELECT 3, S.staff_ID, SL.sale_no, SL.line_total_price
FROM Staff S, Sale SL
WHERE S.staff_ID = 10300 AND SL.sale_no = 33112;

INSERT INTO Invoice (invoice_ID, staff_ID, sale_no, total_price)
SELECT 4, S.staff_ID, SL.sale_no, SL.line_total_price
FROM Staff S, Sale SL
WHERE S.staff_ID = 10100 AND SL.sale_no = 33100;
```

Queries

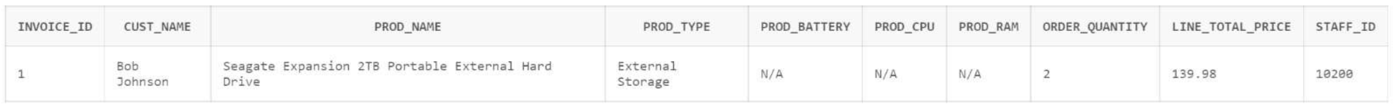
Q1.

--Retrieve data for invoice number one

```
SELECT Invoice.invoice_ID, Customer.cust_name, Product.prod_name, Product.prod_type,
Product.prod_color, Product.prod_CPU, Product.prod_RAM, Sale.order_quantity, Sale.line_total_price,
Staff.staff_ID
FROM Invoice
INNER JOIN Sale ON Invoice.sale_no = Sale.sale_no
INNER JOIN Product ON Sale.prod_ID = Product.prod_ID
INNER JOIN Customer ON Sale.cust_no = Customer.cust_no
INNER JOIN Staff ON Invoice.staff_ID = Staff.staff_ID
WHERE Invoice.invoice_ID = 1;
```



```
SELECT Invoice.invoice_ID, Customer.cust_name, Product.prod_name, Product.prod_type, Product.prod_battery, Product.prod_CPU, Product.prod_RAM, Sale.order_quantity, Sale.line_total_
FROM Invoice
INNER JOIN Sale ON Invoice.sale_no = Sale.sale_no
INNER JOIN Product ON Sale.prod_ID = Product.prod_ID
INNER JOIN Customer ON Sale.cust_no = Customer.cust_no
INNER JOIN Staff ON Invoice.staff_ID = Staff.staff_ID
WHERE Invoice.invoice_ID = 1
```

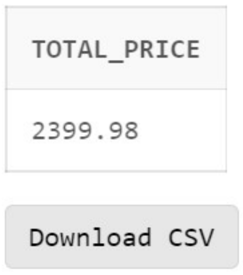


INVOICE_ID	CUST_NAME	PROD_NAME	PROD_TYPE	PROD_BATTERY	PROD_CPU	PROD_RAM	ORDER_QUANTITY	LINE_TOTAL_PRICE	STAFF_ID
1	Bob Johnson	Seagate Expansion 2TB Portable External Hard Drive	External Storage	N/A	N/A	N/A	2	139.98	10200

Q2.

--Retreive the total price of invoice number 2

SELECT SUM (prod_unit_price*order_quantity) AS Total_Price
FROM Sale
INNER JOIN Invoice ON Sale.sale_no = Invoice.sale_no
INNER JOIN Product ON Sale.prod_ID = Product.prod_ID
WHERE Invoice.invoice_ID = 2;



TOTAL_PRICE
2399.98

Conclusion

In conclusion, we have learned how to design and build a database for a retail invoice using SQL Developer and Oracle Database. We have created tables to store information about the items purchased and the invoice details, and we have used SQL queries to manipulate and extract information from the database. We have also gained hands-on experience in using SQL statements to insert, update, and delete data from the tables.

Overall, this coursework has provided us with valuable experience in using SQL Developer and Oracle Database to design and build a simple database for a retail invoice. This knowledge and skills can be applied in a variety of real-world scenarios, such as managing sales and inventory data for a retail business.

Table of Contents