Project Progress Report-3 (Functional Design)

Names:
Farhan Labib, 30176224, farhan.labib@ucalgary.ca
Matthew Yemer, 30047521, matthew.yemer@ucalgary.ca
Tanjim Rahman, 30182328, tanjim.rahman@ucalgary.ca

Table of content:

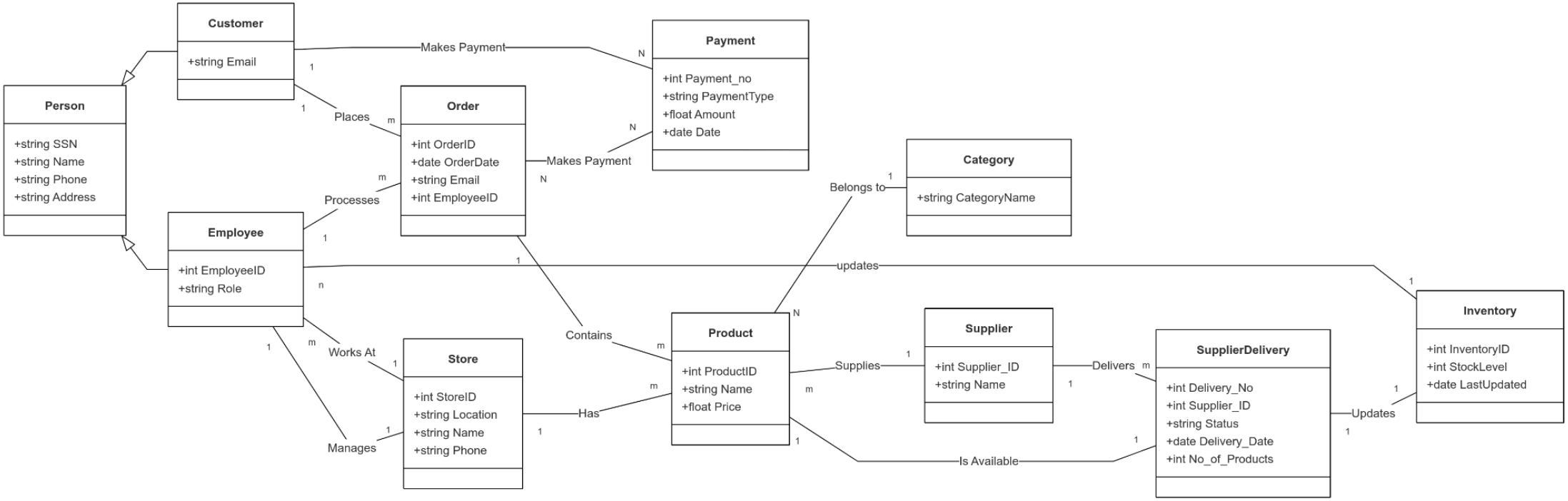
We decided to go with option ii:

- UML Diagram: A class diagram representing the system's structure.

- Sequence Diagrams: Illustrating the complete order placement and processing workflow. Description of the functionality is added in detail.

- SQL statements: Defining the database schema and necessary operations for implementation.

UML Diagram (class diagram):



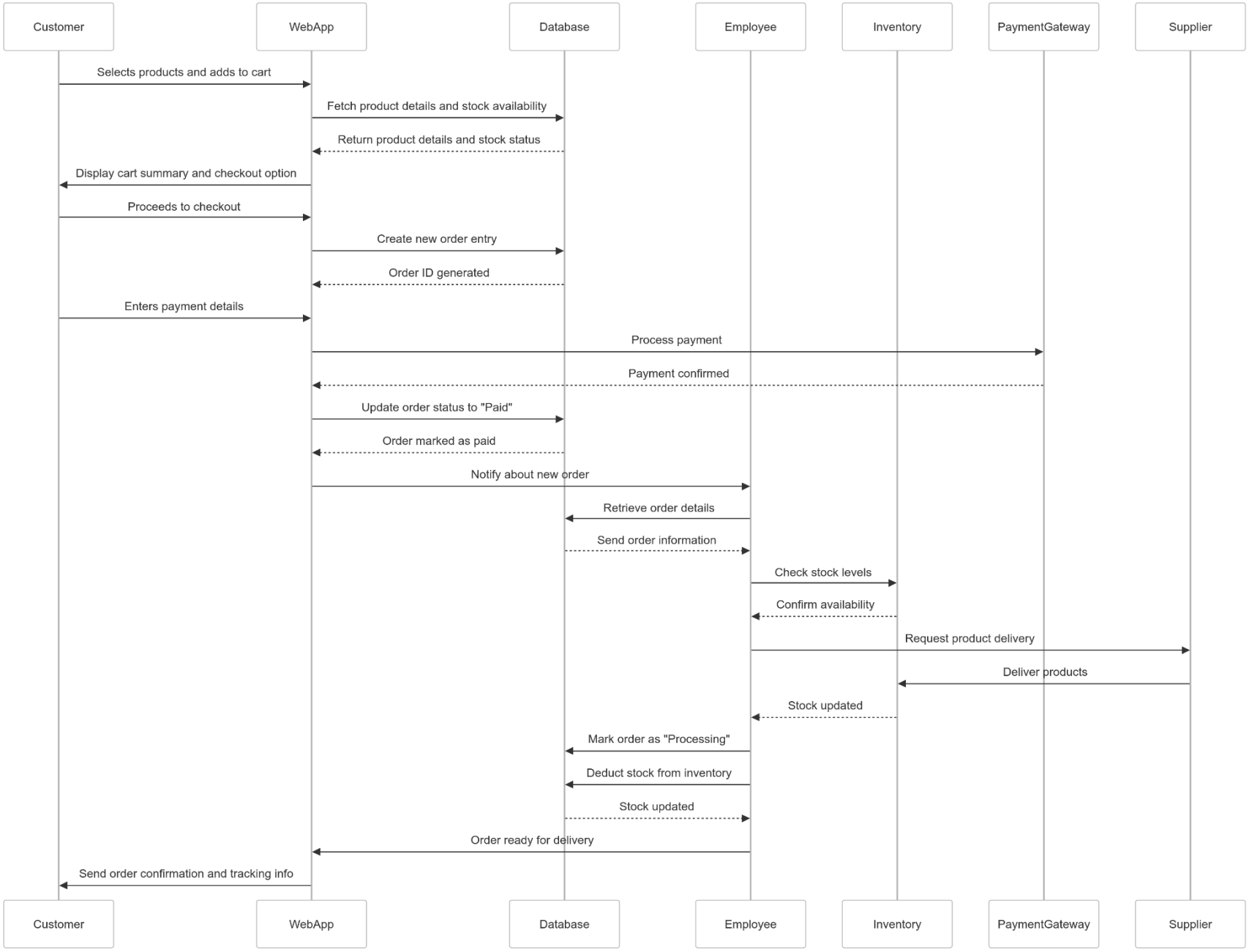
UML Class Diagram Description

The UML class diagram represents the core structure of the Computer Component Store Database System, defining key entities and their relationships.

- Person is a general entity with attributes like SSN, Name, Phone, and Address. It is specialized into Customer and Employee.
- Customer places multiple Orders, which are processed by an Employee.
- Each Order contains multiple Products, which belong to a Category.
- Payments are associated with Orders, ensuring transactions are properly recorded.
- The Store entity manages products and employees. Employees are assigned to a specific store and update the Inventory.
- Suppliers provide Products through Supplier Deliveries, which update the Inventory upon arrival.

This diagram outlines the relationships between these entities, ensuring efficient order processing, inventory management, and supplier coordination.

Sequence Diagram:



Sequence Diagram Description

This diagram outlines the order placement and processing flow in the Computer Component Store System, showing interactions between the Customer, Web App, Database, Employee, Inventory, Payment Gateway, and Supplier

Process Breakdown:

1. **Customer Order Initiation:**
   The Customer selects products and adds them to the cart. The Web App fetches product details and stock status from the Database and displays the cart summary.
2. **Order Placement:**
   The Customer proceeds to checkout. The Web App creates a new order entry in the Database, which generates an Order ID.
3. **Payment Processing:**
   The Customer enters payment details. The Web App processes payment through the Payment Gateway and updates the Database to mark the order as "Paid."
4. **Employee Order Processing:**
   The Web App notifies an Employee about the new order. The Employee retrieves order details and checks stock availability in the Inventory.
5. **Supplier Delivery (If Needed):**
   If stock is low, the Employee requests product delivery from the Supplier. The Inventory updates once products are delivered.
6. **Finalizing Order and Delivery:**
   The Employee updates the order status to "Processing" and deducts stock. The Web App notifies the Customer with order confirmation and tracking info.

SQL on MySQL:

Creating a Database:

```
CREATE DATABASE Computer_Hardware_Store;
```

Creating Tables in the Database:

```
USE Computer_Hardware_Store;

CREATE TABLE STORE(
        STOREID INTEGER PRIMARY KEY,
        LOCATION VARCHAR(255),
        NAME VARCHAR(255) NOT NULL,
        PHONE INTEGER UNIQUE,
        MANAGER_ID INTEGER
        );

CREATE TABLE PERSON(
        SSN INTEGER PRIMARY KEY,
        NAME VARCHAR(255) NOT NULL,
        PHONE INTEGER UNIQUE,
        ADDRESS VARCHAR(255) NOT NULL
        );
```

```
CREATE TABLE CUSTOMER(
        EMAIL VARCHAR(255) PRIMARY KEY,
        SSN INTEGER NOT NULL,
        FOREIGN KEY (SSN) REFERENCES PERSON (SSN)
        );

 CREATE TABLE EMPLOYEE(
        EMPLOYEE_ID INTEGER PRIMARY KEY ,
        SSN INTEGER NOT NULL,
        ROLE VARCHAR(50),
        STOREID INTEGER,
        FOREIGN KEY (SSN) REFERENCES PERSON (SSN),
        FOREIGN KEY (STOREID) REFERENCES STORE (STOREID)
        );

ALTER TABLE STORE
ADD CONSTRAINT fk_manager
FOREIGN KEY (MANAGER_ID) REFERENCES EMPLOYEE (EMPLOYEE_ID);

CREATE TABLE ORDERS(
        ORDER_ID INTEGER PRIMARY KEY ,
        ORDER_DATE DATE NOT NULL,
        EMAIL VARCHAR(255),
        EMPLOYEE_ID INTEGER,
        FOREIGN KEY (EMAIL) REFERENCES CUSTOMER (EMAIL),
        FOREIGN KEY (EMPLOYEE_ID) REFERENCES EMPLOYEE (EMPLOYEE_ID)
        );
```

```
CREATE TABLE CATEGORY(
        CATEGORY_NAME VARCHAR(255) PRIMARY KEY,
        GENERATION INTEGER
        );

CREATE TABLE SUPPLIER(
        SUPPLIER_ID INTEGER PRIMARY KEY,
        NAME VARCHAR(255) NOT NULL
        );

 CREATE TABLE SUPPLIER_DELIVERY(
        DELIVERY_NO INTEGER PRIMARY KEY,
        SUPPLIER_ID INTEGER,
        STATUS VARCHAR(255),
        DELIVERY_DATE DATE NOT NULL,
        FOREIGN KEY (SUPPLIER_ID) REFERENCES SUPPLIER (SUPPLIER_ID)
        );

CREATE TABLE PAYMENT(
        PAYMENT_NO INTEGER PRIMARY KEY ,
        PAYMENT_TYPE VARCHAR(255) NOT NULL,
        AMOUNT INTEGER NOT NULL,
        DATE DATE
        );
```

```
CREATE TABLE PRODUCT(
        PRODUCTID INTEGER PRIMARY KEY ,
        NAME VARCHAR(255),
        PRICE INTEGER,
        ORDER_ID_CONTAINS INTEGER,
        CATEGORY_NAME VARCHAR(255),
        STOREID INTEGER NOT NULL,
        SUPPLIER_ID INTEGER NOT NULL,
        DELIVERY_NO INTEGER NOT NULL,
        NO_OF_PRODUCTS INTEGER,
        FOREIGN KEY (ORDER_ID_CONTAINS) REFERENCES ORDERS (ORDER_ID) ON DELETE CASCADE,
        FOREIGN KEY (CATEGORY_NAME) REFERENCES CATEGORY (CATEGORY_NAME),
        FOREIGN KEY (STOREID) REFERENCES STORE (STOREID),
        FOREIGN KEY (SUPPLIER_ID) REFERENCES SUPPLIER (SUPPLIER_ID),
        FOREIGN KEY (DELIVERY_NO) REFERENCES SUPPLIER_DELIVERY (DELIVERY_NO) ON DELETE CASCADE
        );
```

```
CREATE TABLE INVENTORY(
        INVENTORY_ID INTEGER PRIMARY KEY,
        STOCK_LEVEL INTEGER,
        LAST_UPDATED VARCHAR(255) NOT NULL
        );
```

```
CREATE TABLE MAKES_PAYMENT(
        PAYMENT_NO INTEGER,
        ORDER_ID INTEGER,
        EMAIL VARCHAR(255),
        PRIMARY KEY (PAYMENT_NO, ORDER_ID, EMAIL),
        FOREIGN KEY (ORDER_ID) REFERENCES ORDERS (ORDER_ID) ON DELETE CASCADE,
        FOREIGN KEY (PAYMENT_NO) REFERENCES PAYMENT (PAYMENT_NO),
        FOREIGN KEY (EMAIL) REFERENCES CUSTOMER (EMAIL)
        );


CREATE TABLE UPDATES(
        EMPLOYEE_ID INTEGER,
        INVENTORY_ID INTEGER,
        SUPPLIER_ID INTEGER,
        DELIVERY_NO INTEGER,
        PRIMARY KEY(EMPLOYEE_ID, INVENTORY_ID, SUPPLIER_ID, DELIVERY_NO),
        FOREIGN KEY (EMPLOYEE_ID) REFERENCES EMPLOYEE (EMPLOYEE_ID),
        FOREIGN KEY (INVENTORY_ID) REFERENCES INVENTORY (INVENTORY_ID),
        FOREIGN KEY (SUPPLIER_ID) REFERENCES SUPPLIER (SUPPLIER_ID),
        FOREIGN KEY (DELIVERY_NO) REFERENCES SUPPLIER_DELIVERY (DELIVERY_NO) ON DELETE CASCADE
        );
```

Insert information into a Table

```
INSERT INTO PERSON (SSN, NAME, PHONE, ADDRESS)
VALUES
(54128, 'Jane Doe', 123456789, '12th DoLittle St, Calgary, AB'),
(54529, 'John Smith', 213456789, '2nd Roosevelt Lane, Vancouver, BC'),
(89758, 'Jane Doe', 784596321, 'Jackson Avenue, Salt Lake City, Texas'),
(74159, 'Matthew Masters', 852417936, 'James Earl Jones Boulevard, Astera, NW'),
(96415, 'Farhan Dullahan', 159753248, 'Loc Lac St, Port Tanzania, OW'),
(25479, 'Marjaneh Imani', 300120040, 'Kunafa Tower, Akihabara'),
(98526, 'Sin Cos Tanjim', NULL, 'Grove St, Los Santos, San Andreas');

INSERT INTO CUSTOMER (EMAIL, SSN)
VALUES
('m.masters@gmail.com', 74159),
('jane.doe@hotmail.com', 54128),
('master_inquisitor@ucalgary.ca', 25479),
('drdullahan@ucalgary.ca', 96415);

INSERT INTO STORE (STOREID, LOCATION, NAME, PHONE, MANAGER_ID)
VALUES
(50, 'Castle Schrade, OW', 'Fatalis Corp', 5624189, NULL),
(100, 'Biotechnica Building, Night City', 'Viktor Industries', 74185279, NULL),
(102, NULL, 'NOS', 8524169, NULL),
(148, 'Ides of March Road, Las Venturas, San Andreas', 'Triad', 78963, NULL),
(149, NULL, 'Ice Cold PCs', 74159874, NULL),
(255, '255 University Drive, Calgary, AB', 'Victor Von Doom Hardwares', 123459876, NULL),
(1486, 'Baxter Building', 'Fantastic 4 Inc.', 852741987, NULL);

INSERT INTO EMPLOYEE (EMPLOYEE_ID, SSN, ROLE, STOREID)
VALUES
```

```
(148, 96415, 'Customer Service', 100),
(251, 54529, NULL, 100),
(500, 89758, 'Manager', 102),
(894, 74159, 'Cashier', 1486),
(1000, 54128, 'Manager', 149),
(502, 98526, 'Backroom Loader', 50);

UPDATE STORE
SET MANAGER_ID = 500 WHERE STOREID = 50;

UPDATE STORE
SET MANAGER_ID = 1000 WHERE STOREID = 100;

UPDATE STORE
SET MANAGER_ID = 500 WHERE STOREID = 102;

UPDATE STORE
SET MANAGER_ID = 1000 WHERE STOREID = 148;

UPDATE STORE
SET MANAGER_ID = 500 WHERE STOREID = 149;

UPDATE STORE
SET MANAGER_ID = 1000 WHERE STOREID = 255;

UPDATE STORE
SET MANAGER_ID = 500 WHERE STOREID = 1486;

INSERT INTO ORDERS (ORDER_ID, ORDER_DATE, EMAIL, EMPLOYEE_ID)
VALUES
(100, '2012-01-01', 'jane.doe@hotmail.com', 148),
```

```
(785, '2024-12-28', 'master_inquisitor@ucalgary.ca', 500),
(956, '2021-08-24', 'm.masters@gmail.com', 500),
(5699, '2018-07-13', 'drdullahan@ucalgary.ca', 148),
(7415, '2019-06-19', 'm.masters@gmail.com', 500);

INSERT INTO CATEGORY (CATEGORY_NAME, GENERATION)
VALUES
('Processor', 7),
('Motherboard', 4),
('Laptop', 3),
('Unknown', NULL),
('Nintendo Gaming Console', 5),
('Graphics Card', 7);

INSERT INTO SUPPLIER (SUPPLIER_ID, NAME)
VALUES
(15, 'Arasaka'),
(7, 'Dell'),
(19, 'Guild'),
(78, 'Sentinels'),
(52, 'Gol D. Roger'),
(741, 'AMD');
```

```
INSERT INTO SUPPLIER_DELIVERY (DELIVERY_NO, SUPPLIER_ID, STATUS, DELIVERY_DATE)
VALUES
(2, 741, 'Delivered', '2011-12-12'),
(4, 7, 'Delivered', '2023-12-12'),
(5, 19, 'Delivered', '2025-02-02'),
(24, 15, 'Delivered', '2019-05-19'),
(25, 15, 'Not Delivered', '2020-01-01'),
(41, 78, 'Delivered', '2020-07-24'),
(52, 52, 'Delivered', '2018-07-12');

INSERT INTO PRODUCT (PRODUCTID, NAME, PRICE, ORDER_ID_CONTAINS, CATEGORY_NAME, STOREID, SUPPLIER_ID,
DELIVERY_NO, NO_OF_PRODUCTS)
VALUES
(1, 'Core i7', 999, 7415, 'Processor', 255, 15, 24, 8),
(2, 'RTX 5070', 699, NULL, 'Motherboard', 100, 19, 5, 6),
(15, NULL, NULL, NULL, 'Unknown', 102, 15, 25, 1),
(24, 'Dell Inspiron 3515', 1599, 956, 'Laptop', 1486, 78, 41, 20),
(36, 'Dell Inspiron 3515', 1599, 785, 'Laptop', 148, 7, 4, 10),
(46, 'Nintendo 3DS', 1000, 5699, 'Nintendo Gaming Console', 149, 52, 52, 100),
(50, 'RX 7600XTX', 2000, 100,  'Graphics Card', 50, 741, 2, 1);

INSERT INTO PAYMENT (PAYMENT_NO, PAYMENT_TYPE, AMOUNT, DATE)
VALUES
(1, 'Cash', 999, '2019-06-19'),
(2, 'Debit', 1599, '2021-08-24'),
(3, 'Credit', 1599, '2024-12-28'),
(4, 'Debit', 1000, '2018-07-13'),
(5, 'Debit', 2000, '2012-01-01');
```

```
INSERT INTO INVENTORY (INVENTORY_ID, STOCK_LEVEL, LAST_UPDATED)
VALUES
(1, 8, '2024-03-15'),
(2, 6, '2024-03-14'),
(3, 1, '2024-02-28'),
(4, 20, '2024-01-10'),
(5, 10, '2024-03-01'),
(6, 100, '2024-02-20'),
(7, 1, '2024-03-05');

INSERT INTO MAKES_PAYMENT (PAYMENT_NO, ORDER_ID, EMAIL)
VALUES
(1, 7415, 'm.masters@gmail.com'),
(2, 956, 'm.masters@gmail.com'),
(3, 785, 'master_inquisitor@ucalgary.ca'),
(4, 5699, 'drdullahan@ucalgary.ca'),
(5, 100, 'jane.doe@hotmail.com');

INSERT INTO UPDATES (EMPLOYEE_ID, INVENTORY_ID, SUPPLIER_ID, DELIVERY_NO)
VALUES
(148, 1, 15, 24),
(500, 2, 7, 4),
(894, 3, 19, 5),
(1000, 4, 78, 41),
(502, 5, 52, 52);
```

Select a certain information from table(s):

```
-- Choosing an item to add to order
SELECT 'RTX 5070'
FROM PRODUCT;

-- Finding the most expensive product in the store
SELECT NAME, PRICE, STOREID
FROM PRODUCT
ORDER BY PRICE DESC
LIMIT 1;

-- Find all products supplied by a particular supplier
SELECT P.PRODUCTID, P.NAME, P.PRICE
FROM  PRODUCT AS P
JOIN SUPPLIER AS S ON P.SUPPLIER_ID=S.SUPPLIER_ID
WHERE S.NAME='Arasaka';

-- Find managers and the stores they manage
SELECT E.EMPLOYEE_ID, E.ROLE, S.NAME, S.STOREID
FROM EMPLOYEE AS E
JOIN STORE AS S ON E.EMPLOYEE_ID=S.MANAGER_ID
WHERE E.ROLE='Manager';

-- Find and produce the receipt of a particular order
SELECT MP.PAYMENT_NO, MP.ORDER_ID, MP.EMAIL, P.NAME
FROM MAKES_PAYMENT as MP
JOIN CUSTOMER AS C ON MP.EMAIL=C.EMAIL
```

```
JOIN PERSON AS P ON C.SSN=P.SSN
WHERE P.NAME='Jane Doe';
```

Update Information

```
-- Updating the phone number for a person
UPDATE PERSON
SET PHONE=987654321
WHERE NAME='Jane Doe'
AND SSN=54128;

-- Updating product availability after a delivery so number of available products of a certain kind is now higher.
UPDATE PRODUCT
SET NO_OF_PRODUCTS=20
WHERE PRODUCTID=50;

-- After finding a certain product, we update the order to add to it
UPDATE PRODUCT
SET ORDER_ID_CONTAINS=785
WHERE PRODUCTID=2;


– Updating NULL information for circular dependency

UPDATE STORE
SET MANAGER_ID = 500 WHERE STOREID = 50;

UPDATE STORE
```

```
SET MANAGER_ID = 1000 WHERE STOREID = 100;

UPDATE STORE
SET MANAGER_ID = 500 WHERE STOREID = 102;


UPDATE STORE
SET MANAGER_ID = 1000 WHERE STOREID = 148;

UPDATE STORE
SET MANAGER_ID = 500 WHERE STOREID = 149;

UPDATE STORE
SET MANAGER_ID = 1000 WHERE STOREID = 255;

UPDATE STORE
SET MANAGER_ID = 500 WHERE STOREID = 1486;
```

Delete information from the database:

```
-- Delete a Delivery that will not come and is not delivered
DELETE FROM SUPPLIER_DELIVERY
WHERE STATUS='NOT DELIVERED';

-- Products are too old to sell, so we delete them
DELETE FROM CATEGORY
WHERE GENERATION=3;

-- Delete from inventory after a product is no longer available in the storage inventory
DELETE FROM INVENTORY
WHERE STOCK_LEVEL = 0;

-- Delete a store if they have not provided a location
DELETE FROM STORE
WHERE LOCATION IS NULL
AND STOREID IN(
SELECT DISTINCT STOREID
FROM PRODUCT
WHERE STOREID IS NOT NULL
);
```