Project 3

Ahmed Tlili, Leon Petrinos, Mathilde Peruzzo

March 25, 2025

Relational Schema

- **Store**(s_id, s_address, phone_number, manager_id UNIQUE NOT NULL)
  FOREIGN KEY(manager_id) REFERENCES Employee(employee_id)
- **Employee**(e_id, e_name, s_id)
  FOREIGN KEY(s_id) REFERENCES Store(s_id)
- **Manufacturer**(m_id, m_name)
- **Product**(p_id, p_name NOT NULL, unit_price NOT NULL, description, discount_percentage, m_id NOT NULL)
  FOREIGN KEY(m_id) REFERENCES Manufacturer(m_id)
- **Paint**(p_id, base, color)
  FOREIGN KEY(p_id) REFERENCES Product(p_id)
- **Tool**(p_id, type)
- **Has_in_stock**(p_id, s_id, quantity NOT NULL)
  FOREIGN KEY(p_id) REFERENCES Product(p_id)
  FOREIGN KEY(s_id) REFERENCES Store(s_id)
- **Customer**(email, c_name, c_address NOT NULL)
  PRIMARY KEY(email)
- **Purchase**(p_id, amount NOT NULL, p_date NOT NULL, p_time NOT NULL)
- **Contains_purchase**(p_id, product_id, quantity NOT NULL)
  FOREIGN KEY(p_id) REFERENCES Purchase(p_id)
  FOREIGN KEY(product_id) REFERENCES Product(p_id)
- **Instore**(p_id, e_id)
  FOREIGN KEY(p_id) REFERENCES Purchase(p_id)
  FOREIGN KEY(e_id) REFERENCES Employee(e_id)
- **Online**(p_id, rating, delivery_fee NOT NULL, email NOT NULL)
  FOREIGN KEY(p_id) REFERENCES Purchase(p_id)
  FOREIGN KEY(email) REFERENCES Customer(email)

Stored Procedure

(a) This stored procedure increases the discount of products that have never been sold by 10% until the maximum discount is reached. Maximum discount is the input argument. If the current discount is already greater or equal to the maximum discount, the discount will not be increased. If when increasing the current discount, the maximum discount is reached surpassed, the discount will be set to the maximum discount (no more). Otherwise, the current discount will be increased by 10%.

(b)

```
CREATE OR REPLACE PROCEDURE DiscountInactiveProducts(IN
    max_discount INT)
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE current_pid INT;
    DECLARE current_discount INT;

    DECLARE product_cursor CURSOR FOR
        SELECT p_id, COALESCE(discount_pourcentage, 0)
        FROM Product
        WHERE COALESCE(discount_pourcentage, 0) < max_discount;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN product_cursor;

    FETCH product_cursor INTO current_pid, current_discount;

    WHILE done = 0 DO

        IF (NOT EXISTS (
            SELECT 1
            FROM Contains_purchase
            WHERE product_id = current_pid)
        )
        THEN
            IF (current_discount + 10 > max_discount) THEN
                SET current_discount = max_discount;
            ELSE
                SET current_discount = current_discount + 10;
            END IF;

            UPDATE Product
            SET discount_pourcentage = current_discount
            WHERE p_id = current_pid;
        END IF;

        FETCH product_cursor INTO current_pid, current_discount;

    END WHILE;

    CLOSE product_cursor;
END
```

(c)

Calling the procedure with maximum discount of 25%

```
db2 => CALL DiscountInactiveProducts(25);

  Return Status = 0
```

First 25 products (id and discount) before calling procedure:

```
db2 => SELECT p_id, discount_pourcentage FROM Product WHERE p_id <= 25;

P_ID        DISCOUNT_POURCENTAGE
----------- --------------------
          1                   10
          2                   15
          3                    5
          4                   20
          5                   10
          6                    0
          7                   25
          8                   30
          9                   12
         10                    5
         11                   15
         12                   20
         13                    8
         14                   10
         15                   18
         16                    7
         17                   22
         18                   10
         19                   15
         20                   12
         21                    9
         22                   20
         23                    5
         24                   30
         25                    5

  25 record(s) selected.
```

First 25 products (id and discount) after calling procedure. Note that the products with id 1 to 20 have purchases, so only the discounts from 21 to 25 change.

```
db2 => SELECT p_id, discount_pourcentage FROM Product WHERE p_id <= 25;

P_ID        DISCOUNT_POURCENTAGE
----------- --------------------
          1                   10
          2                   15
          3                    5
          4                   20
          5                   10
          6                    0
          7                   25
          8                   30
          9                   12
         10                    5
         11                   15
         12                   20
         13                    8
         14                   10
         15                   18
         16                    7
         17                   22
         18                   10
         19                   15
         20                   12
         21                   19
         22                   25
         23                   15
         24                   30
         25                   15

  25 record(s) selected.
```

Application Program

The application handles all the errors gracefully.
This is the current menu of the application.

```
Paint Store Management System
        1. List sales generated by every store
        2. Add new Employee
        3. Search for a product by name substring
        4. Change the discount percentage of a product
        5. Issue refund for online purchase
        6. Biggest purchase in every store
        7. Quit
Please Enter Your Option:
```

Let's discuss what each option does:

- Type 1 to select Option 1: list sales generated by every store

```
Paint Store Management System
        1. List sales generated by every store
        2. Add new Employee
        3. Search for a product by name substring
        4. Change the discount percentage of a product
        5. Issue refund for online purchase
        6. Biggest purchase in every store
        7. Quit
Please Enter Your Option: 1
```

Output is the following:

```
Store ID  Address              Total Amount
---------------------------------------
1         9301 Leopard St           44.47
2         456 Oakwood Dr           195.90
3         789 Maple Ave            157.41
4         321 Pine St              279.86
5         654 Cedar Ln             162.39
6         987 Birch Rd               0.00
7         159 Elm St                 0.00
8         753 Walnut Blvd            0.00
---------------------------------------
Press Enter to Continue...
```

- Type 2 to select Option 2: Add new Employee

```
Paint Store Management System
        1. List sales generated by every store
        2. Add new Employee
        3. Search for a product by name substring
        4. Change the discount percentage of a product
        5. Issue refund for online purchase
        6. Biggest purchase in every store
        7. Quit
Please Enter Your Option: 2
```

Just enter the name of the employee and the store he will be working for, output is the following:

```
Enter the full name: Ahmed Tlili
Enter the store ID: 2
Employee added successfully!
 Given Employee ID: 52
Press Enter to Continue...
```

- Type 3 to select Option 3: Search for a product by name substring

```
Paint Store Management System
        1. List sales generated by every store
        2. Add new Employee
        3. Search for a product by name substring
        4. Change the discount percentage of a product
        5. Issue refund for online purchase
        6. Biggest purchase in every store
        7. Quit
Please Enter Your Option: 3
```

Example below shows the output when searching for the products with name containing "Peach"

```
Enter the product name: Peach
Product ID          Product Name        Unit Price          Description                                                Discount
-------------------------------------------------------------------------------------------------------------------------------
22                  Peach Blush         16.99               Warm peach paint for living rooms                          25
46                  Peach Fizz          13.49               Fresh peach paint for warm living spaces                   18
88                  Pale Peach          16.49               Light peach color for dining rooms                         22
-------------------------------------------------------------------------------------------------------------------------------
Press Enter to Continue...
```

- Type 4 to select Option 4: Change the discount percentatge of a product

```
Paint Store Management System
        1. List sales generated by every store
        2. Add new Employee
        3. Search for a product by name substring
        4. Change the discount percentage of a product
        5. Issue refund for online purchase
        6. Biggest purchase in every store
        7. Quit
Please Enter Your Option: 4
```

Just enter the ID of the product and the new discount percentage, output is the following:

```
Enter the product ID: 5
Enter the new discount percentage: 10
Discount percentage updated successfully!
Press Enter to Continue...
```

- Type 5 to select Option 5: Issue refund for online purchase

```
Paint Store Management System
        1. List sales generated by every store
        2. Add new Employee
        3. Search for a product by name substring
        4. Change the discount percentage of a product
        5. Issue refund for online purchase
        6. Biggest purchase in every store
        7. Quit
Please Enter Your Option: 5
```

Just enter the email of the customer, a list of all his online purchases will be displayed, then you can select the purchase you want to refund by typing the correct number in the submenu. Output is the following:

```
Enter the Customer email: michael.wilson@example.com
Purchase IDAmount    Date                Time
------------------------------------------------------------
7         88.45     2025-03-07          12:25:00
------------------------------------------------------------
Enter the Purchase ID to refund: 7
Refund issued successfully!
Press Enter to Continue...
```

- Type 6 to select Option 6: Biggest purchase in every store

```
Paint Store Management System
        1. List sales generated by every store
        2. Add new Employee
        3. Search for a product by name substring
        4. Change the discount percentage of a product
        5. Issue refund for online purchase
        6. Biggest purchase in every store
        7. Quit
Please Enter Your Option: 6
```

Output is the following:

```
Store ID  Address              Total Amount
----------------------------------------
1         9301 Leopard St          44.47
2         456 Oakwood Dr          107.45
3         789 Maple Ave            90.95
4         321 Pine St              97.45
5         654 Cedar Ln            105.93
6         987 Birch Rd              0.00
7         159 Elm St                0.00
8         753 Walnut Blvd           0.00
----------------------------------------
Press Enter to Continue...
```

- Type 7 to select Option 7: Quit

```
Paint Store Management System
        1. List sales generated by every store
        2. Add new Employee
        3. Search for a product by name substring
        4. Change the discount percentage of a product
        5. Issue refund for online purchase
        6. Biggest purchase in every store
        7. Quit
Please Enter Your Option: 7
```

Correctly exits the application.

Indexing

Index 1

(a)
```
db2 => CREATE INDEX clustered_purchase_idx ON Purchase(p_date) CLUSTER;
DB20000I  The SQL command completed successfully.
```

(b) A clustered index on purchase date in the Purchase table is beneficial because purchases are frequently analyed based on dates and date ranges. Thus, sorting the purchases by date allows for efficient range queries, making it faster to access data for accounting purposes. An example query that would benefit from this index is the following:

```
SELECT SUM(amount) AS total
FROM Purchase
WHERE p_date >= '01/01/2025' AND p_date <= '12/31/2025';
```

This above query computes the total revenue for the year 2025. With this clustered index, the database can quickly locate the first matching row, and perform a sequential scan to retrieve all rows within the specified date range, without needing to follow the pointers of other data entries (value + rid), as in a non-clustered index, which could often leed to more IO.

Index 2

(a)
```
db2 => CREATE INDEX stock_idx ON Has_in_stock(s_id, quantity) CLUSTER;
DB20000I  The SQL command completed successfully.
```

(b) This index is on the s_id and quantity attribute of the Has_in_stock table. It is useful for this application to efficiently identify products that are running low in stock in a specific store, which is crucial for inventory management. An example query that would benefit from this index is the following:

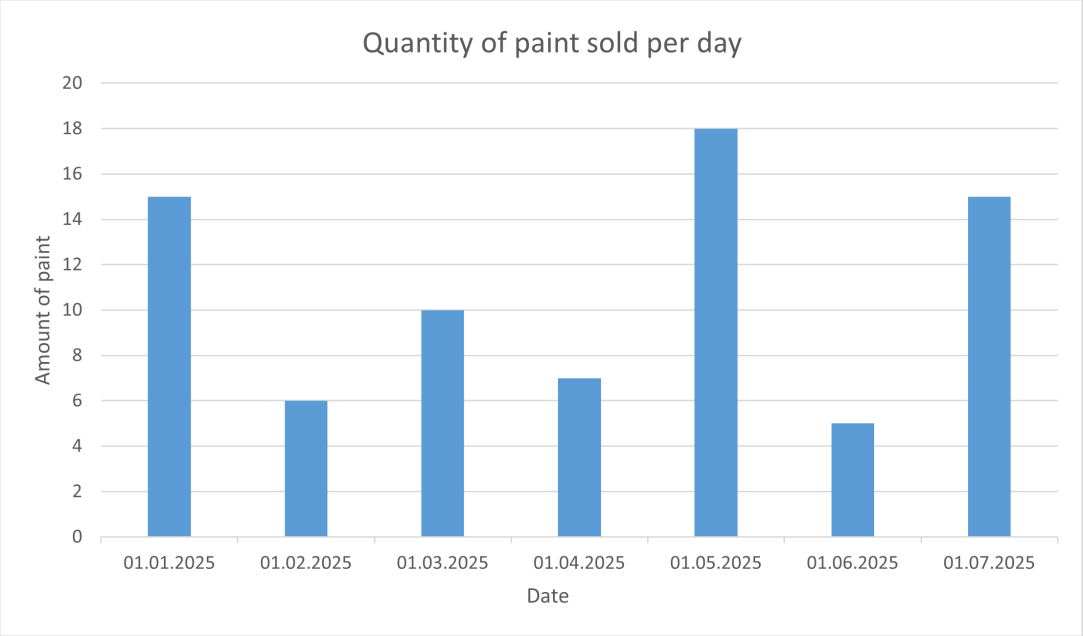
```
SELECT p_id
FROM Has_in_stock
WHERE s_id = 0 AND quantity < 5;
```

This query identifies all products of a particular store where the quantity of the product is very limited. The fact that it is a clustered index, again, allows for efficient range queries, making it faster to access data for inventory management purposes. It also makes sense to use a clustered index because the other attributes of the table are id's, which are certainly not needed in a sorted order.

Visualisation

Vis 1

The following chart represents the daily amount of paint sold during the first 7 days of March (01.03.2025 - 07.03.2025).

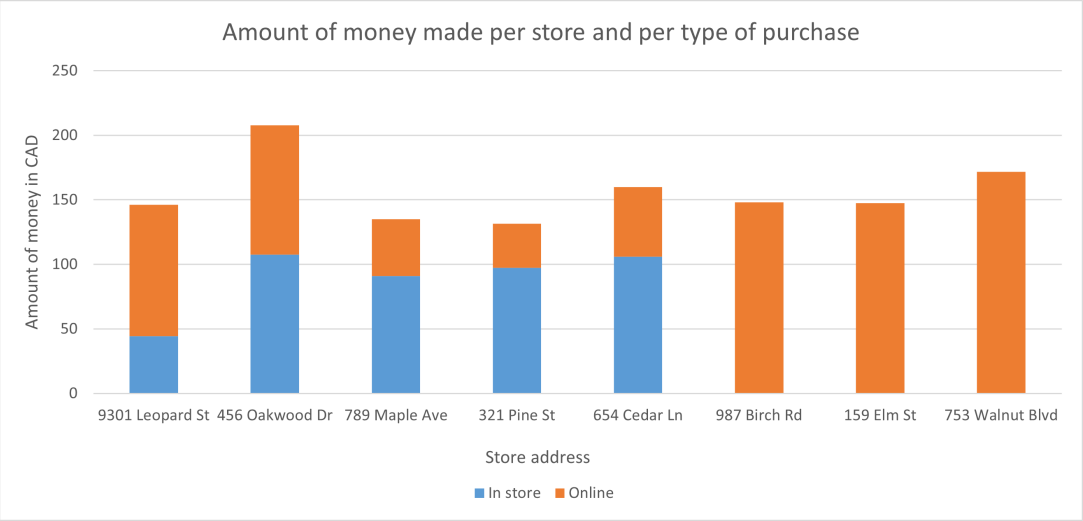


The query used to find the data :

```
 SELECT PURCHASE.p_date, SUM(CONTAINS_PURCHASE.quantity)
FROM PAINT
JOIN CONTAINS_PURCHASE ON CONTAINS_PURCHASE.product_id = PAINT.p_id
JOIN PURCHASE ON CONTAINS_PURCHASE.purchase_id = PURCHASE.p_id
GROUP BY PURCHASE.p_date
HAVING '2025-03-01' <= PURCHASE.p_date
AND PURCHASE.p_date <= '2025-03-07';
```

Vis 2

The following chart represent the amount of money made by each store for online and in-store purchases during the first 7 days of March (01.03.2025 - 07.03.2025).



The query used to find the data :

```
 WITH TEMP_INSTORE AS (
SELECT STORE.s_id, s_address, COALESCE(SUM(amount), 0) AS total_amount
FROM STORE
```

```
LEFT JOIN EMPLOYEE ON STORE.s_id = EMPLOYEE.s_id
LEFT JOIN INSTORE ON EMPLOYEE.e_id = INSTORE.e_id
LEFT JOIN PURCHASE ON INSTORE.p_id = PURCHASE.p_id AND '2025-03-01' <= PURCHASE.p_date
AND PURCHASE.p_date <= '2025-03-07'
GROUP BY STORE.s_id , s_address
),
TEMP_ONLINE AS (
SELECT STORE.s_id, s_address, COALESCE(SUM(amount), 0) AS total_amount
FROM STORE
LEFT JOIN ONLINE ON STORE.s_id = ONLINE.s_id
LEFT JOIN PURCHASE ON ONLINE.p_id = PURCHASE.p_id AND '2025-03-01' <= PURCHASE.p_date
AND PURCHASE.p_date <= '2025-03-07'
GROUP BY STORE.s_id, s_address
)
SELECT TEMP_INSTORE.s_id, TEMP_INSTORE.s_address,
TEMP_INSTORE.total_amount AS instore_amount, TEMP_ONLINE.total_amount AS online_amount
FROM TEMP_INSTORE
LEFT JOIN TEMP_ONLINE ON TEMP_INSTORE.s_id = TEMP_ONLINE.s_id
ORDER BY TEMP_INSTORE.s_id ASC;
```

Creativity

a) For this section, we chose to write a trigger. The trigger gives a 10% discount on online purchases for clients that have already made at least 10 online purchases in the last 30 days.

b)

```
CREATE OR REPLACE TRIGGER eligibleForDiscount AFTER INSERT ON ONLINE
REFERENCING NEW AS n
FOR EACH ROW
BEGIN
DECLARE date_cutoff DATE;
DECLARE count INTEGER;

SELECT PURCHASE.p_date - 30 DAYS
INTO date_cutoff
FROM PURCHASE
WHERE PURCHASE.p_id = n.p_id;

SELECT COUNT(*)
INTO count
FROM ONLINE JOIN PURCHASE
ON ONLINE.p_id = PURCHASE.p_id AND email = n.email AND date_cutoff <= PURCHASE.p_date;

IF (count > 10)
THEN
UPDATE PURCHASE
SET amount = 0.9 * amount
WHERE PURCHASE.p_id = n.p_id;
END IF;
END;
```

c) The client Lisa Brown with email (primary key) "lisa.brown@example.com" has already made 10 online purchase in the month preceding 2025-03-07. The insertion of the corresponding entries in the database is not shown here.

We insert the entries for Brown's 11th online purchase on 2025-03-07 and query the amount for this purchase :

```
db2 => INSERT INTO PURCHASE (p_id, amount, p_date, p_time) VALUES (20, 153.81, '2025-03-07', '18:20:00');
DB20000I  The SQL command completed successfully.
db2 => INSERT INTO CONTAINS_PURCHASE (purchase_id, product_id, quantity) VALUES (20, 52, 9);
DB20000I  The SQL command completed successfully.
db2 => INSERT INTO ONLINE (p_id, rating, delivery_fee, email, s_id) VALUES (20, NULL, 0, 'lisa.brown@example.com', 8);
DB20000I  The SQL command completed successfully.
db2 => SELECT PURCHASE.amount FROM PURCHASE WHERE PURCHASE.p_id = 20;

AMOUNT
-------
 138.42

  1 record(s) selected.
```

The amount returned by the query (138.42 CAD) corresponds to 90% of the amount inserted (153.81 CAD) as expected.

Work Division

We had two meetings to discuss the project and the work division. We decided to divide the work as follows:

- Ahmed Tlili: Question 4 coding the application program
- Leon Petrinos: Question 3, 5 about indexing and stored procedures
- Mathilde Peruzzo: Question 6, 7 about visualisation and creativity