**1. Generate missing transaction dates for each campaign using a FOR loop (PostgreSQL)**

**Problem:**  
If a campaign runs from start\_date to end\_date, generate all missing dates in between.

sql

CopyEdit

DO $$

DECLARE

campaign\_start DATE := '2024-01-01'; -- Example start date

campaign\_end DATE := '2024-01-10'; -- Example end date

current\_date DATE;

BEGIN

current\_date := campaign\_start;

WHILE current\_date <= campaign\_end LOOP

INSERT INTO missing\_dates (date) VALUES (current\_date);

current\_date := current\_date + INTERVAL '1 day';

END LOOP;

END $$;

**2. Find all customers who have used the same coupon at least N times using a WHILE loop (MySQL)**

**Problem:**  
Loop through customers and count coupon redemptions.

sql

CopyEdit

DELIMITER //

CREATE PROCEDURE CheckCouponUsage(IN min\_redemptions INT)

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE customer\_id\_var INT;

DECLARE coupon\_count INT;

DECLARE cur CURSOR FOR

SELECT customer\_id, COUNT(\*) FROM train WHERE redemption\_status = 1 GROUP BY customer\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: WHILE NOT done DO

FETCH cur INTO customer\_id\_var, coupon\_count;

IF coupon\_count >= min\_redemptions THEN

INSERT INTO high\_usage\_customers (customer\_id, coupon\_redemptions) VALUES (customer\_id\_var, coupon\_count);

END IF;

END WHILE;

CLOSE cur;

END //

DELIMITER ;

**3. Assign a loyalty rank to customers based on coupon redemptions using a FOR loop (PL/pgSQL)**

**Problem:**  
Classify customers into **Gold, Silver, or Bronze** based on the number of coupons redeemed.

sql

CopyEdit

DO $$

DECLARE

rec RECORD;

BEGIN

FOR rec IN (SELECT customer\_id, COUNT(\*) AS total\_redemptions FROM train WHERE redemption\_status = 1 GROUP BY customer\_id) LOOP

IF rec.total\_redemptions >= 10 THEN

INSERT INTO customer\_loyalty (customer\_id, loyalty\_level) VALUES (rec.customer\_id, 'Gold');

ELSIF rec.total\_redemptions BETWEEN 5 AND 9 THEN

INSERT INTO customer\_loyalty (customer\_id, loyalty\_level) VALUES (rec.customer\_id, 'Silver');

ELSE

INSERT INTO customer\_loyalty (customer\_id, loyalty\_level) VALUES (rec.customer\_id, 'Bronze');

END IF;

END LOOP;

END $$;

**4. Generate a sequence of campaign IDs using a FOR loop (PL/SQL in Oracle)**

**Problem:**  
Insert campaign IDs into a table for future tracking.

sql

CopyEdit

DECLARE

i NUMBER := 1;

BEGIN

FOR i IN 1..10 LOOP

INSERT INTO campaign\_tracking (campaign\_id, status) VALUES (i, 'Active');

END LOOP;

END;

**5. Find customers who have consecutively redeemed coupons using a WHILE loop (MySQL)**

**Problem:**  
Check **consecutive redemptions** across campaigns.

sql

CopyEdit

DELIMITER //

CREATE PROCEDURE ConsecutiveRedemptions()

BEGIN

DECLARE prev\_customer INT DEFAULT NULL;

DECLARE curr\_customer INT;

DECLARE curr\_campaign INT;

DECLARE prev\_campaign INT DEFAULT NULL;

DECLARE done INT DEFAULT FALSE;

DECLARE cur CURSOR FOR

SELECT customer\_id, campaign\_id FROM train WHERE redemption\_status = 1 ORDER BY customer\_id, campaign\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

WHILE NOT done DO

FETCH cur INTO curr\_customer, curr\_campaign;

IF curr\_customer = prev\_customer AND curr\_campaign = prev\_campaign + 1 THEN

INSERT INTO repeat\_redeemers (customer\_id, campaign\_id) VALUES (curr\_customer, curr\_campaign);

END IF;

SET prev\_customer = curr\_customer;

SET prev\_campaign = curr\_campaign;

END WHILE;

CLOSE cur;

END //

DELIMITER ;

**6. Iterate over campaigns and find the redemption rate using a FOR loop (PL/pgSQL)**

**Problem:**  
Loop over all campaigns and calculate the **redemption rate**.

sql

CopyEdit

DO $$

DECLARE

rec RECORD;

redemption\_rate NUMERIC;

BEGIN

FOR rec IN (SELECT campaign\_id, COUNT(\*) AS total, SUM(CASE WHEN redemption\_status = 1 THEN 1 ELSE 0 END) AS redeemed FROM train GROUP BY campaign\_id) LOOP

redemption\_rate := (rec.redeemed \* 100.0 / rec.total);

INSERT INTO campaign\_performance (campaign\_id, redemption\_rate) VALUES (rec.campaign\_id, redemption\_rate);

END LOOP;

END $$;

**7. Count unique coupons redeemed per customer using a WHILE loop (MySQL)**

**Problem:**  
Find how many unique coupons each customer has redeemed.

sql

CopyEdit

DELIMITER //

CREATE PROCEDURE CountUniqueCoupons()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE customer\_id\_var INT;

DECLARE coupon\_count INT;

DECLARE cur CURSOR FOR

SELECT customer\_id, COUNT(DISTINCT coupon\_id) FROM train WHERE redemption\_status = 1 GROUP BY customer\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

WHILE NOT done DO

FETCH cur INTO customer\_id\_var, coupon\_count;

INSERT INTO customer\_coupon\_count (customer\_id, unique\_coupons) VALUES (customer\_id\_var, coupon\_count);

END WHILE;

CLOSE cur;

END //

DELIMITER ;

**8. Simulate a promotional campaign by inserting random customers into a test table (PL/pgSQL)**

**Problem:**  
Randomly select customers for a campaign.

sql

CopyEdit

DO $$

DECLARE

rec RECORD;

rand\_customer INT;

BEGIN

FOR rec IN 1..10 LOOP

SELECT customer\_id INTO rand\_customer FROM train ORDER BY RANDOM() LIMIT 1;

INSERT INTO promotional\_campaign (customer\_id, campaign\_id) VALUES (rand\_customer, 1001);

END LOOP;

END $$;

**9. Assign a progressive discount to customers using a FOR loop (PL/SQL in Oracle)**

**Problem:**  
Give **progressively larger discounts** to customers based on past redemptions.

sql

CopyEdit

DECLARE

rec\_customer INT;

rec\_redemptions INT;

BEGIN

FOR rec\_customer, rec\_redemptions IN (SELECT customer\_id, COUNT(\*) FROM train WHERE redemption\_status = 1 GROUP BY customer\_id) LOOP

INSERT INTO customer\_discounts (customer\_id, discount)

VALUES (rec\_customer, rec\_redemptions \* 5); -- 5% extra discount per redemption

END LOOP;

END;

**10. Simulate coupon expiration by removing expired coupons using a WHILE loop (MySQL)**

**Problem:**  
Remove expired coupons from the active table.

sql

CopyEdit

DELIMITER //

CREATE PROCEDURE ExpireCoupons()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE exp\_coupon INT;

DECLARE cur CURSOR FOR

SELECT coupon\_id FROM campaign\_data WHERE end\_date < CURDATE();

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

WHILE NOT done DO

FETCH cur INTO exp\_coupon;

DELETE FROM active\_coupons WHERE coupon\_id = exp\_coupon;

END WHILE;

CLOSE cur;

END //

DELIMITER ;