Function which counts the number of records

1. general counting

```
CREATE OR REPLACE FUNCTION count_records
  RETURN NUMBER
IS
  v_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_count FROM shipping;
  RETURN v_count;
END;

select count_records() from dual;
```

2. counting card users

```
CREATE OR REPLACE FUNCTION count_cardusers
RETURN NUMBER
IS
  v_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_count FROM payment WHERE payment_method
<> 'Cash';
  RETURN v_count;
END;

select count_cardusers() from dual;
```

3. counting books with loss of money

```
CREATE OR REPLACE FUNCTION count_loss
RETURN NUMBER
IS
  v_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_count FROM inventory WHERE purchase_price
> sell_price;
  RETURN v_count;
END;

select count_loss() from dual;
```

4. counting old books

```
CREATE OR REPLACE FUNCTION count_old_books
RETURN NUMBER
IS
  v_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_count FROM book WHERE publication_date <
```

```
TO_DATE('2000-01-01',
'YYYY-MM-DD');
  RETURN v_count;
END;

select count_old_books() from dual;
```

5. free delivery for orders with total price > 100

```
CREATE OR REPLACE FUNCTION count_free_delivery
RETURN NUMBER
IS
  v_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_count FROM orders WHERE total_price >=
100.00;
  RETURN v_count;
END;

select count_free_delivery() from dual;
```

Procedure which uses SQL%ROWCOUNT to determine the number of rows affected

1. update the price: discount

```
CREATE OR REPLACE PROCEDURE discount IS
  row_count NUMBER;
BEGIN
  UPDATE shopping_cart SET total_price = total_price * 0.95 WHERE customer_id =
'3';
  row_count := SQL%ROWCOUNT;
  DBMS_OUTPUT.PUT_LINE('Number of rows affected: ' ll row_count);
END;
/

BEGIN
discount;
END;
```

2. update the price: cancel the discount

```
CREATE OR REPLACE PROCEDURE cancel_discount IS
  row_count NUMBER;
BEGIN
```

```
  UPDATE shopping_cart SET total_price = total_price * 100/95 WHERE
customer_id = '3';
  row_count := SQL%ROWCOUNT;
  DBMS_OUTPUT.PUT_LINE('Number of rows affected: ' ll row_count);
END;
/

BEGIN
cancel_discount;
END;
```

3. delete old customer

VER 1

```
CREATE OR REPLACE PROCEDURE delete_customer IS
  row_count NUMBER;
BEGIN
  DELETE FROM customer WHERE customer_id = '11';
  row_count := SQL%ROWCOUNT;
  DBMS_OUTPUT.PUT_LINE('Number of rows affected: ' ll row_count);
END;
/
```

VER 2

```
CREATE OR REPLACE PROCEDURE delete_customer IS
  row_count NUMBER;
BEGIN
  DELETE FROM customer WHERE address LIKE  '%Kaskelen%';
  row_count := SQL%ROWCOUNT;
  DBMS_OUTPUT.PUT_LINE('Number of rows affected: ' ll row_count);
END;
/

begin
delete_customer;
end;
```

4. insert new customers

```
CREATE OR REPLACE PROCEDURE new_customers IS
  row_count NUMBER;
BEGIN
  INSERT INTO customer (customer_id, first_name, last_name, email, address,
phone_number)
  VALUES ('11', 'Zhaniya', 'Abubakirova', 'zhaniya@example.com', '3 Altyn Aul,
Kaskelen, Kz', '555-9146' );
  INSERT INTO customer (customer_id, first_name, last_name, email, address,
```

```
phone_number)
  VALUES ('12', 'Asel', 'Nazar', 'aselnazar@example.com', '22 Altyn Aul, Kaskelen,
Kz', '555-9100' );
  row_count := SQL%ROWCOUNT;
  DBMS_OUTPUT.PUT_LINE('Number of rows affected: ' || row_count);
END;
/

begin
new_customers;
end;
```