Emag

1. Резултати от създадените функции, приложени върху данните от база данни Emag_DB:

- Функция PRODUCT_AVERAGE_RATING, която извежда среден рейтинг за даден продукт

```
CREATE OR ALTER FUNCTION PRODUCT_AVERAGE_RATING(@PRODUCT_ID
INT)
RETURNS VARCHAR(200)
AS
BEGIN
  DECLARE @PROD_NAME VARCHAR(100), @AVG_RATING NUMERIC(4,2);


  SELECT @PROD_NAME = NAME
  FROM PRODUCT
  WHERE PRODUCT_ID = @PRODUCT_ID;


  SELECT @AVG_RATING = ISNULL(AVG(CAST(RATING AS DECIMAL(4,2))),
0.00)
  FROM REVIEW
  WHERE PRODUCT_ID = @PRODUCT_ID;

  RETURN ISNULL(@PROD_NAME, 'Непознат продукт')
    + ' (ID = ' + CAST(@PRODUCT_ID AS VARCHAR) + '), среден рейтинг: '
    + CAST(@AVG_RATING AS VARCHAR) + '.';
END
GO

SELECT dbo.PRODUCT_AVERAGE_RATING(PRODUCT_ID) AS
PRODUCT_AVERAGE_RATING_RESULT
FROM PRODUCT;
```

	PRODUCT_AVERAGE_RATING_RESULT
1	iPhone 14 (ID = 2), среден рейтинг: 2.50.
2	iPhone 14 Pro (ID = 3), среден рейтинг: 4.33.
3	iPhone 15 (ID = 4), среден рейтинг: 4.50.
4	Galaxy S23 (ID = 5), среден рейтинг: 5.00.
5	Galaxy S23 Ultra (ID = 6), среден рейтинг: 4.00.
6	Galaxy A55 (ID = 7), среден рейтинг: 3.00.
7	IdeaPad 3 (ID = 8), среден рейтинг: 3.50.
8	ThinkPad X1 (ID = 9), среден рейтинг: 5.00.
9	Legion 5 Pro (ID = 10), среден рейтинг: 3.00.
10	XPS 13 (ID = 11), среден рейтинг: 2.50.
11	Inspiron 15 (ID = 12), среден рейтинг: 5.00.
12	WH-1000XM5 (ID = 13), среден рейтинг: 4.50.
13	WF-C700N (ID = 14), среден рейтинг: 4.50.
14	LinkBuds S (ID = 15), среден рейтинг: 4.50.
15	EOS M50 (ID = 16), среден рейтинг: 3.50.
16	EOS R10 (ID = 17), среден рейтинг: 4.50.
17	PowerShot G7X (ID = 18), среден рейтинг: 5.00.
18	Nintendo Switch (ID = 19), среден рейтинг: 4.00.
19	Nintendo Switch OLED (ID = 20), среден рейти...
20	PlayStation 5 (ID = 21), среден рейтинг: 4.00.
21	Air Fryer 5L (ID = 22), среден рейтинг: 3.00.
22	Blender 700W (ID = 23), среден рейтинг: 3.50.

Фигура 1. Резултат от функцията PRODUCT_AVERAGE_RATING.

- Функция SELLER_AVERAGE_RATING, която извежда среден рейтинг за даден продавач (SELLER).

```
CREATE OR ALTER FUNCTION SELLER_AVERAGE_RATING(@SELLER_ID INT)
RETURNS VARCHAR(200)
AS
BEGIN
  DECLARE @SELLER_NAME VARCHAR(100), @AVG_RATING NUMERIC(4,2);

  -- Взимаме името на фирмата
  SELECT @SELLER_NAME = COMPANY_NAME
  FROM SELLER
  WHERE SELLER_ID = @SELLER_ID;

  -- Изчисляваме средния рейтинг по всички продукти на този продавач
  SELECT @AVG_RATING = ISNULL(AVG(CAST(R.RATING AS DECIMAL(4,2))),
0.00)
  FROM PRODUCT P
  JOIN REVIEW R ON R.PRODUCT_ID = P.PRODUCT_ID
  WHERE P.SELLER_ID = @SELLER_ID;

  RETURN ISNULL(@SELLER_NAME, 'Непознат продавач')
    + ' (ID = ' + CAST(@SELLER_ID AS VARCHAR) + '), среден рейтинг: '
    + CAST(@AVG_RATING AS VARCHAR) + '.';
```

```
END
GO

SELECT dbo.SELLER_AVERAGE_RATING(SELLER_ID) AS
SELLER_AVERAGE_RATING_RESULT
FROM SELLER;
```

	SELLER_AVERAGE_RATING_RESULT
1	HomeKitch (ID = 8), среден рейтинг: 3.60.
2	AutoGear (ID = 15), среден рейтинг: 4.27.
3	Lenovo Outlet (ID = 3), среден рейтинг: 3.83.
4	Dell Store (ID = 4), среден рейтинг: 3.75.
5	ToyBox (ID = 14), среден рейтинг: 3.60.
6	BeautyCare (ID = 10), среден рейтинг: 4.33.
7	GameHub (ID = 7), среден рейтинг: 4.38.
8	CleanHome (ID = 12), среден рейтинг: 4.60.
9	Galaxy Traders (ID = 2), среден рейтинг: 3.83.
10	PhotoPro (ID = 6), среден рейтинг: 4.33.
11	FashionLine (ID = 11), среден рейтинг: 3.80.
12	TechNova Ltd (ID = 1), среден рейтинг: 3.86.
13	SportStar (ID = 13), среден рейтинг: 4.75.
14	Audio World (ID = 5), среден рейтинг: 4.11.
15	Book Planet (ID = 9), среден рейтинг: 4.13.

Фигура 2. Резултат от функцията SELLER_AVERAGE_RATING.

2. Резултати от създадените съхранени процедури, приложени върху данните от база данни Emag_DB:

- Съхранена процедура USER_ORDERS, Процедура, която извежда броя поръчки и общо похарчена сума по потребители.

```
CREATE PROCEDURE USER_ORDERS
AS
SELECT
   U.USERNAME,
   U.CITY,
   COUNT(O.ORDER_ID)              AS ORDER_COUNT,
   CAST(ISNULL(SUM(O.TOTAL_AMOUNT), 0.00) AS DECIMAL(12,2)) AS
TOTAL_SPENT
FROM [USER] U
LEFT JOIN [ORDER] O
   ON O.USER_ID = U.USER_ID
```

```
GROUP BY U.USERNAME, U.CITY
ORDER BY TOTAL_SPENT DESC, ORDER_COUNT DESC;

EXEC USER_ORDERS;
```

	USERNAME	CITY	ORDER_COUNT	TOTAL_SPENT
1	ivan123	Sofia	2	3440.00
2	georgi_t	Varna	3	2919.98
3	petar_k	Ruse	3	2569.97
4	iva_t	Plovdiv	3	2554.98
5	todor_m	Sofia	2	2510.00
6	viktor_s	Varna	3	2354.97
7	vesela_k	Varna	3	1939.98
8	maria_pop	Plovdiv	3	1879.97
9	yordan_n	Burgas	2	1860.00
10	nikolay_p	Pleven	2	1820.00
11	stefka_v	Sofia	3	1819.96
12	elena_m	Stara Zagora	3	1689.98
13	ani_d	Burgas	3	1539.97
14	kristina_a	Pleven	3	1344.97
15	martin_g	Sofia	2	1180.00

Фигура 3. Резултат от съхранената процедура USER_ORDERS.

- Съхранена процедура SELLER_TOP_PRODUCTS, която извежда продуктите на продавач с бройки и оборот, подредени по оборот.

```
CREATE PROCEDURE SELLER_TOP_PRODUCTS
   @SELLER_ID INT
AS
SELECT
   P.PRODUCT_ID,
   P.NAME              AS PRODUCT_NAME,
   SUM(OP.QUANTITY)       AS TOTAL_QTY,
   CAST(SUM(OP.QUANTITY * OP.UNIT_PRICE) AS DECIMAL(18,2)) AS REVENUE
FROM PRODUCT P
JOIN ORDER_PRODUCT OP
   ON OP.PRODUCT_ID = P.PRODUCT_ID
WHERE P.SELLER_ID = @SELLER_ID
GROUP BY P.PRODUCT_ID, P.NAME
ORDER BY REVENUE DESC, TOTAL_QTY DESC;
```

```
EXEC SELLER_TOP_PRODUCTS 1;
```

	PRODUCT_ID	PRODUCT_NAME	TOTAL_QTY	REVENUE
1	2	iPhone 14	3	2450.00
2	4	iPhone 15	2	1499.99
3	3	iPhone 14 Pro	1	950.00

Фигура 4. Резултат от съхранената процедура SELLER_TOP_PRODUCTS.

- Съхранена процедура ORDERS_IN_PERIOD, която извежда поръчките в период.

```
CREATE PROCEDURE ORDERS_IN_PERIOD
   @DateFrom DATE,
   @DateTo   DATE
AS
SELECT
   o.ORDER_ID,
   o.ORDER_NUMBER,
   o.ORDER_DATE,
   u.USERNAME,
   o.STATUS,
   -- тотал от детайлите (по-надежден от записания)
   CAST(SUM(op.QUANTITY * op.UNIT_PRICE) AS DECIMAL(18,2)) AS
CalculatedTotal,
   -- платено по поръчка
   CAST(ISNULL( (SELECT SUM(p.AMOUNT) FROM PAYMENT p WHERE
p.ORDER_ID = o.ORDER_ID), 0.00) AS DECIMAL(18,2)) AS PaidAmount,
   -- баланс
   CAST(
      SUM(op.QUANTITY * op.UNIT_PRICE)
      - ISNULL( (SELECT SUM(p.AMOUNT) FROM PAYMENT p WHERE
p.ORDER_ID = o.ORDER_ID), 0.00)
      AS DECIMAL(18,2)
   ) AS Balance
FROM [ORDER] o
JOIN [USER] u      ON u.USER_ID = o.USER_ID
JOIN ORDER_PRODUCT op ON op.ORDER_ID = o.ORDER_ID
WHERE o.ORDER_DATE >= @DateFrom
 AND o.ORDER_DATE <  DATEADD(DAY, 1, @DateTo)  -- правим @DateTo
включително
GROUP BY o.ORDER_ID, o.ORDER_NUMBER, o.ORDER_DATE, u.USERNAME,
o.STATUS
```

```
ORDER BY o.ORDER_DATE DESC, o.ORDER_ID DESC;
GO

EXEC ORDERS_IN_PERIOD '2024-10-01','2025-10-31';
```

	ORDER_ID	ORDER_NUMBER	ORDER_DATE	USERNAME	STATUS	CalculatedTotal	PaidAmount	Balance
1	40	ORD-2025-0040	2025-03-12	viktor_s	Cancelled	144.97	144.97	0.00
2	39	ORD-2025-0039	2025-03-11	elena_m	Delivered	519.98	519.98	0.00
3	38	ORD-2025-0038	2025-03-10	ani_d	Delivered	139.97	139.97	0.00
4	37	ORD-2025-0037	2025-03-09	maria_pop	Pending	309.97	309.97	0.00
5	36	ORD-2025-0036	2025-03-08	vesela_k	Delivered	169.98	169.98	0.00
6	35	ORD-2025-0035	2025-03-07	kristina_a	Delivered	114.97	114.97	0.00
7	34	ORD-2025-0034	2025-03-06	iva_t	Shipped	224.98	224.98	0.00
8	33	ORD-2025-0033	2025-03-05	stefka_v	Delivered	429.96	429.96	0.00
9	32	ORD-2025-0032	2025-03-04	petar_k	Delivered	779.97	779.97	0.00
10	31	ORD-2025-0031	2025-03-03	georgi_t	Delivered	1079.98	1079.98	0.00
11	30	ORD-2025-0030	2025-03-02	todor_m	Delivered	1010.00	1010.00	0.00
12	29	ORD-2025-0029	2025-03-01	vesela_k	Delivered	880.00	880.00	0.00
13	28	ORD-2025-0028	2025-02-28	yordan_n	Shipped	540.00	540.00	0.00
14	27	ORD-2025-0027	2025-02-27	kristina_a	Delivered	460.00	460.00	0.00
15	26	ORD-2025-0026	2025-02-26	martin_g	Delivered	620.00	620.00	0.00
16	25	ORD-2025-0025	2025-02-25	iva_t	Pending	1350.00	1350.00	0.00

Фигура 5. Резултат от съхранената процедура ORDERS_IN_PERIOD.

3. Резултати от създадените тригери, приложени върху данните от база данни Emag_DB:

- Тригер: преизчислява TOTAL_AMOUNT на поръчка при промени по ORDER_PRODUCT.
- Работи при INSERT, UPDATE и DELETE. Поправя и NULL UNIT_PRICE с PRODUCT.PRICE.

```
CREATE OR ALTER TRIGGER trg_OrderProduct_RecalcOrderTotal
ON ORDER_PRODUCT
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
  SET NOCOUNT ON;

  -- 1) Поправи UNIT_PRICE, ако е NULL (за ново/обновено)
  IF EXISTS (SELECT 1 FROM inserted WHERE UNIT_PRICE IS NULL)
  BEGIN
    UPDATE op
    SET UNIT_PRICE = p.PRICE
    FROM ORDER_PRODUCT op
```

```
      JOIN inserted i  ON i.ORDER_ID = op.ORDER_ID AND i.PRODUCT_ID =
op.PRODUCT_ID
      JOIN PRODUCT  p  ON p.PRODUCT_ID = op.PRODUCT_ID
      WHERE op.UNIT_PRICE IS NULL;
  END

  -- 2) Намери засегнатите поръчки
  ;WITH affected AS (
    SELECT ORDER_ID FROM inserted
    UNION
    SELECT ORDER_ID FROM deleted
  )
  -- 3) Пресметни новия тотал от редовете (Quantity * Unit_Price)
  UPDATE o
  SET TOTAL_AMOUNT = ISNULL((
        SELECT CAST(SUM(op.QUANTITY * op.UNIT_PRICE) AS DECIMAL(10,2))
        FROM ORDER_PRODUCT op
        WHERE op.ORDER_ID = o.ORDER_ID
      ), 0.00)
  FROM [ORDER] o
  JOIN affected a ON a.ORDER_ID = o.ORDER_ID;
END
GO
```

- trg_UpdateSellerAverageRating

```
CREATE OR ALTER TRIGGER trg_UpdateSellerAverageRating
ON dbo.REVIEW
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
  SET NOCOUNT ON;

  DECLARE @SellerId INT;

  -- Определи кой търговец е засегнат (взимаме го от продукта)
  SELECT TOP 1 @SellerId = p.SELLER_ID
  FROM dbo.PRODUCT p
  JOIN (
    SELECT PRODUCT_ID FROM inserted
    UNION
    SELECT PRODUCT_ID FROM deleted
  ) x ON x.PRODUCT_ID = p.PRODUCT_ID;
```

```
    -- Ако няма засегнат търговец, излизаме
    IF @SellerId IS NULL
        RETURN;

    -- Пресмятаме новия среден рейтинг за този търговец
    DECLARE @NewAvg DECIMAL(4,2);

    SELECT @NewAvg = AVG(CAST(r.RATING AS DECIMAL(4,2)))
    FROM dbo.REVIEW r
    JOIN dbo.PRODUCT pr ON pr.PRODUCT_ID = r.PRODUCT_ID
    WHERE pr.SELLER_ID = @SellerId;

    -- Обновяваме средната оценка (ако няма ревюта, става 0)
    UPDATE dbo.SELLER
    SET AVERAGE_RATING = ISNULL(@NewAvg, 0.00)
    WHERE SELLER_ID = @SellerId;
END;
GO

-- Пример: добавяме ред към поръчка без UNIT_PRICE (ще вземе
PRODUCT.PRICE)
INSERT INTO ORDER_PRODUCT (ORDER_ID, PRODUCT_ID, QUANTITY,
UNIT_PRICE)
VALUES (1, 1, 2, NULL);

-- Проверка: тоталът се е актуализирал
SELECT ORDER_ID, TOTAL_AMOUNT FROM [ORDER] WHERE ORDER_ID = 1;

-- Обновяване на количество -> тоталът пак се актуализира
UPDATE ORDER_PRODUCT
SET QUANTITY = QUANTITY + 1
WHERE ORDER_ID = 1 AND PRODUCT_ID = 1;

SELECT ORDER_ID, TOTAL_AMOUNT FROM [ORDER] WHERE ORDER_ID = 1;

-- Изтриване на ред -> тоталът се преизчислява
DELETE FROM ORDER_PRODUCT
WHERE ORDER_ID = 1 AND PRODUCT_ID = 1;

SELECT ORDER_ID, TOTAL_AMOUNT FROM [ORDER] WHERE ORDER_ID = 1;
```