Temat: System do obsługi stacji benzynowej

Autorzy:

Rafał Dubiel 25 Daniel Antolak 25 Filip Adamus 25 Filip Przepiórka 25

1. Zakres i krótki opis systemu

Projekt ma na celu stworzenie systemu do obsługi procesu tankowania na stacji benzynowej. W projektowanej bazie danych będą rejestrowane wszelkie informacje, wymagane do poprawnego funkcjonowania stacji. Wszystkie rekordy będą odpowiednio zapisywane w utworzonych tabelach pod unikalnymi wierszami. Dodatko w systemie zostaną zaimplementowanych funkcje, widoki, procedury, mające na celu ułatwnienie czynności związanych z obsługą stacji oraz prowadzeniem analizy danych.

Klient przyjeżdża na stacje paliw, wybiera odpowiedni, czynny dystrybutor i rodzaj paliwa, które chce zatankować. Wybranie nieczynnego dystrybutora, uniemożliwi kontynuowanie procesu. Podnosi pistolet i zaczyna proces tankowania. Po zakończeniu czynności, musi odłożyć pistolet na swoje miejsce. Bez zakończenia procesu, klient nie może zapłacić za zatankowane paliwo. Odłożenie pistoletu jest wyznacznikiem zakończenia tankowania. Po odłożeniu, klient może przystąpić do płacenia. Po opłaceniu rachunku przez klienta, zapisana zostanie kwota transakcji, przyznana zniżka , którą klient może posiadać, data, numer dystrybutora z którego pobrano paliwo, pracownika który obsłużył zamówienie oraz numer samochodu. Klient ma możliwość zakupu paliwa na paragon lub fakturę. Do wzięcia faktury system wymaga wprowadzenia numeru NIP. Klient może podać numery rejestracyjne pojazdu, nie jest to jednak wymogiem.

System będzie pozwalał na manipulacje cenami paliwa przez pracowników, otrzymanych od członka zarządu stacji. Zmiany cen paliwa są rejestrowane w bazie wraz z datami, w których te zmiany zostały wprowadzone.

System przechowywać będzie informacje na temat dostawców paliwa, dystrybutorów, klientów, transakcji oraz pracowników stacji. Będzie możliwość generowania raportów oraz zarządzania transakcjami. System będzie dostępny dla pracowników stacji paliw oraz właścicieli.

2. Wymagania i funkcje systemu

System powinien rejestrować wszystkie informacje dotyczące sprzedaży paliwa, klientów, pracowników, dostaw, rozliczeń.

System powinien zawierać odpowiednie procedury, widoki i funkcje służące organizacji, analizie i zarządzaniu bazą danych.

System powinien umożliwiać zapisanie danych firmy do faktury poprzez unikalny numer NIP, aby przyspieszyć proces przy następnej sprzedaży.

Jako klient, chcę mieć możliwość otrzymania faktury lub paragonu za kupione paliwo.

Jako klient, chcę mieć możliwość wyboru co do podawania numeru rejestracyjnego do faktury.

Jako sprzedawca chcę mieć możliwość uzyskania raportu sprzedaży z konkretnego dnia.

System powinien rejestrować dostawy paliwa wraz z konkretną datą, ceną, ilością i typem paliwa oraz dostawcą.

System powinien rejestrować jaki typ paliwa jest tankowany.

System powinien dać możliwość wygenerowania wszystkich informacji z konkretnych sprzedaży.

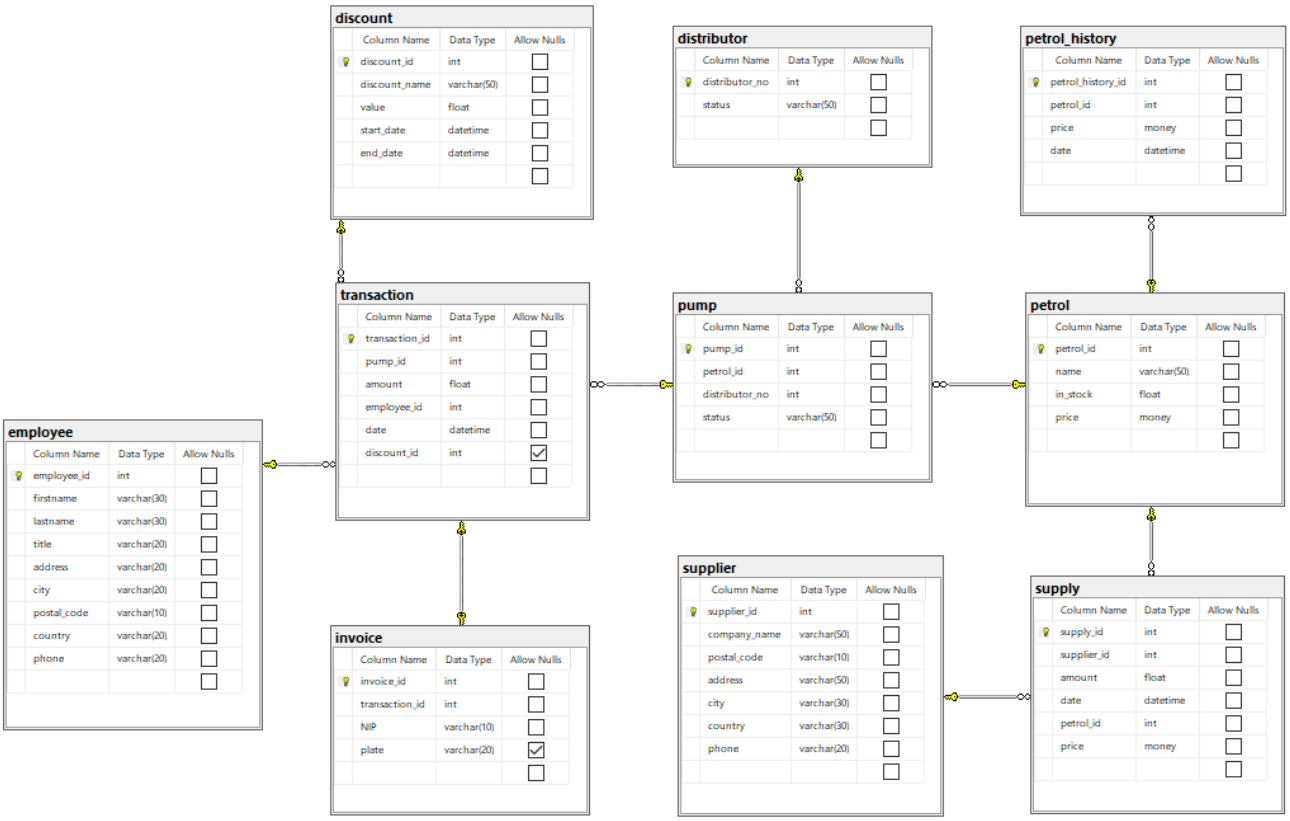
System powinien umożliwić monitorowanie zapasów paliwa.

System powinien rejestrować zmiany cen paliwa.

Jako pracownik, chcę wiedzieć jaka cena paliwa była w określonym dniu.

3. Projekt bazy danych

Diagram przedstawiający schemat bazy danych



Opis poszczególnych tabel

Nazwa tabeli: petrol_history - Opis: Tabela przechowująca historię cen paliwa.

Nazwa atrybutu	Typ	Opis/Uwagi
petrol_history_id	int	PK, Numer ID historii paliwa
petrol_id	int	FK do tabeli petrol, Numer ID paliwa
price	money	Historyczna cena paliwa za litr
date	datetime	Data

Nazwa tabeli: petrol

- Opis: Tabela zawierająca informacje o paliwach.

Nazwa atrybutu	Typ	Opis/Uwagi
petrol_id	int	PK, Numer ID paliwa
name	varchar(50)	Nazwa paliwa
in_stock	float	Aktualny stan paliwa w litrach
price	money	Aktualna cena paliwa za litr

Nazwa tabeli: supplier

- Opis: Tabela zawierająca dokładne dane dostawców paliwa.

Nazwa atrybutu	Typ	Opis/Uwagi
supplier_id	int	PK, Numer ID dostawcy
company_name	varchar(50)	Nazwa dostawcy
postal_code	varchar(10)	Kod pocztowy
address	varchar(50)	Adres siedziby firmy
city	varchar(30)	Miasto dostawcy
country	varchar(30)	Państwo dostawcy
phone	varchar(20)	Numer telefonu dostawcy

Nazwa tabeli: supply

- Opis: Tabela zawierająca dane na temat dostaw.

Nazwa atrybutu	Typ	Opis/Uwagi
supply_id	int	PK, Numer ID Dostawy
supplier_id	int	FK do tabeli supplier, Numer ID dostawcy
amount	float	Ilość dostarczonego paliwa w litrach
date	datetime	Data dostawy
petrol_id	int	FK do tabeli petrol, Numer ID paliwa
price	money	Cena paliwa za litr

Nazwa tabeli: pump

- Opis: Tabela reprezentująca konkretne pistolety paliwa oraz ich dystrybutory.

Nazwa atrybutu	Typ	Opis/Uwagi
pump_id	int	PK, Numer ID pistoletu
petrol_id	int	FK do tabeli petrol, Numer ID paliwa
distributor_no	int	Numer dystrybutora
status	varchar(50)	Status pompy

Nazwa tabeli: transaction

- Opis: Tabela zawierające informacje o transakcjach kupna.

Nazwa atrybutu	Typ	Opis/Uwagi
transaction_id	int	PK, Numer ID transakcji
pump_id	int	FK do tabeli pump, Numer ID pistoletu
amount	float	Ilość zakupionego paliwa
employee_id	int	FK do tabeli employee, Numer ID pracownika
date	datetime	Data transakcji
discount_id	int	Numer ID rabatu (zezwala na wartość NULL)

Nazwa tabeli: employee

- Opis: Tabela zawierająca informacje o pracownikach.

Nazwa atrybutu	Typ	Opis/Uwagi
employee_id	int	PK, Numer ID pracownika
firstname	varchar(30)	Imię pracownika
lastname	varchar(30)	Nazwisko pracownika
title	varchar(20)	Stanowisko pracownika
address	varchar(20)	Adres pracownika
city	varchar(20)	Miasto pracownika
postal_code	varchar(10)	Kod pocztowy
country	varchar(20)	Państwo pracownika
phone	varchar(20)	Numer telefonu

Nazwa tabeli: invoice

- Opis: Tabela zawierająca dane faktur.

Nazwa atrybutu	Typ	Opis/Uwagi
invoice_id	int	PK, Numer ID faktury
transaction_id	int	FK do tabeli transaction, Numer ID transakcji
NIP	varchar(10)	Numer NIP firmy
plate	varchar(20)	Numer rejestracyjny (zezwala na wartość NULL)

Nazwa tabeli: discount

- Opis: Tabela zawierająca specyfikację rabatów.

Nazwa atrybutu	Typ	Opis/Uwagi
discount_id	int	PK, Numer ID rabatu
discount_name	varchar(50)	Nazwa rabatu
value	float	Wartość rabatu
start_date	datetime	Data rozpoczęcia zniżki
end_date	datetime	Data zakończenia zniżki

Nazwa tabeli: distributor

- Opis: Tabela zawierająca informację o dystrybutorach.

Nazwa atrybutu	Typ	Opis/Uwagi
distributor_no	int	PK, Numer dystrybutora
status	varchar(50)	Status dystrybutora

4. Implementacja

Kod poleceń DDL

(dla każdej tabeli należy wkleić kod DDL polecenia tworzącego tabelę)

Tabela: discount

```
CREATE TABLE dbo.discount (
    [discount_id] [int] IDENTITY(4,1) NOT NULL,
    [discount_name] [varchar](50) NOT NULL,
    [value] [float] NOT NULL,
    [start_date] [datetime] NOT NULL,
    [end_date] [datetime] NOT NULL,
    CONSTRAINT CK_discount_value
        CHECK ([value] > 0 AND [value] <= 1),
    CONSTRAINT CK_discount_dates
        CHECK ([start_date] < [end_date]),
    CONSTRAINT PK_discount PRIMARY KEY CLUSTERED (discount_id ASC)
) ON [PRIMARY];
```

Tabela: distributor

```
CREATE TABLE dbo.distributor (
    distributor_no INT NOT NULL,
    status VARCHAR(50) NOT NULL,
    CONSTRAINT CK_distributor_no
        CHECK ([distributor_no] > 0),
    CONSTRAINT PK_distributor PRIMARY KEY CLUSTERED (distributor_no ASC)
) ON [PRIMARY];
```

Tabela: employee

```
CREATE TABLE dbo.employee (
    employee_id INT NOT NULL,
    firstname VARCHAR(30) NOT NULL,
    lastname VARCHAR(30) NOT NULL,
    title VARCHAR(20) NOT NULL,
    address VARCHAR(20) NOT NULL,
    city VARCHAR(20) NOT NULL,
    postal_code VARCHAR(10) NOT NULL,
    country VARCHAR(20) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    CONSTRAINT CK_employee_id
        CHECK (employee_id > 0),
    CONSTRAINT PK_employee PRIMARY KEY CLUSTERED (employee_id ASC)
) ON [PRIMARY];
```

Tabela: invoice

```
CREATE TABLE dbo.invoice (
    invoice_id INT NOT NULL,
    transaction_id INT NOT NULL,
    NIP VARCHAR(10) NOT NULL,
    plate VARCHAR(20) NULL,
    CONSTRAINT CK_transaction_id
        CHECK (transaction_id > 0),
    CONSTRAINT CK_invoice_id
        CHECK (invoice_id > 0),
    CONSTRAINT CK_invoice_NIP
        CHECK (NIP LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
    CONSTRAINT PK_invoice PRIMARY KEY CLUSTERED (invoice_id ASC)
) ON [PRIMARY];

ALTER TABLE dbo.invoice WITH CHECK ADD CONSTRAINT FK_invoice_transaction FOREIGN
KEY (transaction_id)
REFERENCES dbo.transaction (transaction_id);

ALTER TABLE dbo.invoice CHECK CONSTRAINT FK_invoice_transaction;
```

Tabela: petrol

```
CREATE TABLE dbo.petrol (
    petrol_id INT NOT NULL,
    name VARCHAR(50) NOT NULL,
    in_stock FLOAT NOT NULL,
    price MONEY NOT NULL,
    CONSTRAINT CK_petrol_id
        CHECK (petrol_id > 0),
    CONSTRAINT CK_petrol_price
        CHECK (price > 0),
    CONSTRAINT PK_petrol PRIMARY KEY CLUSTERED (petrol_id ASC)
) ON [PRIMARY];
```

Tabela: petrol_history

```
CREATE TABLE dbo.petrol_history (
    petrol_history_id INT NOT NULL,
    petrol_id INT NOT NULL,
    price MONEY NOT NULL,
    [date] DATETIME NOT NULL,
    CONSTRAINT CK_petrol_history_id
        CHECK (petrol_history_id > 0),
    CONSTRAINT CK_petrol_history_price_positive
        CHECK (price > 0),
 CONSTRAINT PK_petrol_history PRIMARY KEY CLUSTERED (petrol_history_id ASC)
) ON [PRIMARY];

ALTER TABLE dbo.petrol_history  WITH CHECK ADD CONSTRAINT FK_petrol_history_petrol
FOREIGN KEY(petrol_id)
REFERENCES dbo.petrol (petrol_id)

ALTER TABLE dbo.petrol_history CHECK CONSTRAINT FK_petrol_history_petrol
```

Tabela: pump

```
CREATE TABLE dbo.pump (
    pump_id INT NOT NULL,
    petrol_id INT NOT NULL,
    distributor_no INT NOT NULL,
    status VARCHAR(50) NOT NULL,
    CONSTRAINT CK_pump_id
        CHECK (pump_id > 0),
    CONSTRAINT CK_pump_petrol_id
        CHECK (petrol_id > 0),
    CONSTRAINT CK_pump_distributor_no_positive
        CHECK (distributor_no > 0),
    CONSTRAINT PK_pump PRIMARY KEY CLUSTERED (pump_id ASC)
) ON [PRIMARY];

ALTER TABLE dbo.pump WITH CHECK ADD CONSTRAINT FK_pump_distributor FOREIGN
KEY(distributor_no)
REFERENCES dbo.distributor (distributor_no);

ALTER TABLE dbo.pump CHECK CONSTRAINT FK_pump_distributor;

ALTER TABLE dbo.pump WITH CHECK ADD CONSTRAINT FK_pump_petrol FOREIGN
KEY(petrol_id)
REFERENCES dbo.petrol (petrol_id);

ALTER TABLE dbo.pump CHECK CONSTRAINT FK_pump_petrol;
```

Tabela: supplier

```
CREATE TABLE dbo.supplier (
    supplier_id INT NOT NULL,
    company_name VARCHAR(50) NOT NULL,
    postal_code VARCHAR(10) NOT NULL,
    address VARCHAR(50) NOT NULL,
    city VARCHAR(30) NOT NULL,
    country VARCHAR(30) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    CONSTRAINT CK_supplier_id
        CHECK (supplier_id > 0),
    CONSTRAINT PK_supplier PRIMARY KEY CLUSTERED (supplier_id ASC)
) ON [PRIMARY];
```

Tabela: supply

```
CREATE TABLE dbo.supply (
    supply_id INT NOT NULL,
    supplier_id INT NOT NULL,
    amount FLOAT NOT NULL,
    [date] DATETIME NOT NULL,
    petrol_id INT NOT NULL,
    price MONEY NOT NULL,
    CONSTRAINT CK_supply_id
        CHECK (supply_id > 0),
    CONSTRAINT CK_supply_price
        CHECK (price > 0),
    CONSTRAINT CK_supply_supplierid
        CHECK (supplier_id > 0),
    CONSTRAINT PK_supply PRIMARY KEY CLUSTERED (supply_id ASC)
) ON [PRIMARY];

ALTER TABLE dbo.supply WITH CHECK ADD CONSTRAINT FK_supply_petrol FOREIGN
KEY(petrol_id)
REFERENCES dbo.petrol (petrol_id);

ALTER TABLE dbo.supply CHECK CONSTRAINT FK_supply_petrol;

ALTER TABLE dbo.supply WITH CHECK ADD CONSTRAINT FK_supply_supplier FOREIGN
KEY(supplier_id)
REFERENCES dbo.supplier (supplier_id);

ALTER TABLE dbo.supply CHECK CONSTRAINT FK_supply_supplier;
```

Tabela: transaction

```
CREATE TABLE dbo.transaction (
    transaction_id INT NOT NULL,
    pump_id INT NOT NULL,
    amount FLOAT NOT NULL,
    employee_id INT NOT NULL,
    [date] DATETIME NOT NULL,
    discount_id INT NULL,
    CONSTRAINT CK_transaction_amount
        CHECK (amount > 0),
    CONSTRAINT PK_transaction PRIMARY KEY CLUSTERED (transaction_id ASC)
) ON [PRIMARY];

ALTER TABLE dbo.transaction WITH CHECK ADD CONSTRAINT FK_transaction_discount
FOREIGN KEY(discount_id)
REFERENCES dbo.discount (discount_id);

ALTER TABLE dbo.transaction CHECK CONSTRAINT FK_transaction_discount;

ALTER TABLE dbo.transaction WITH CHECK ADD CONSTRAINT FK_transaction_Employee
FOREIGN KEY(employee_id)
REFERENCES dbo.employee (employee_id);

ALTER TABLE dbo.transaction CHECK CONSTRAINT FK_transaction_Employee;

ALTER TABLE dbo.transaction WITH CHECK ADD CONSTRAINT FK_transaction_pump1 FOREIGN
KEY(pump_id)
REFERENCES dbo.pump (pump_id);

ALTER TABLE dbo.transaction CHECK CONSTRAINT FK_transaction_pump1;
```

Widoki

(dla każdego widoku należy wkleić kod polecenia definiującego widok wraz z komentarzem)

1. Wartość całkowita każdej transakcji

Widok "vw_bill_value" służy do wyliczenia i zaprezentowania całkowitej wartości każdej transakcji klienta. Uwzględnione zostały również takie wytyczne jak numer transakcji czy ilość zatankowanego paliwa. W cenę uwzględniona jest zniżka w miejscach gdzie ona widnieje. (FP)

```
CREATE VIEW vw_bill_value AS
SELECT
  t.transaction_id,
  t.pump_id,
  t.employee_id,
  p.name,
  t.amount,
  ph.price,
  d.value as discount,
  ROUND(t.amount * ph.price *
  CASE
  WHEN d.value IS NULL THEN 1
  ELSE (1-d.value)
  END
  ,2) as bill_value
FROM [transaction] t
LEFT JOIN discount d ON d.discount_id = t.discount_id
JOIN pump ON pump.pump_id = t.pump_id
JOIN petrol p ON p.petrol_id = pump.petrol_id
JOIN petrol_history ph on ph.petrol_id = p.petrol_id
AND ph.date = (
    select MAX(date)
    from petrol_history
    where date <= t.date)
```

	transaction_id	pump_id	employee_id	name	amount	price	discount	bill_value
1	19	1	2	PB 95	19,02	5,45	NULL	103,66
2	27	1	3	PB 95	23,13	5,48	NULL	126,75
3	30	1	2	PB 95	29,69	5,48	0,05	154,57
4	31	1	2	PB 95	31,23	5,48	NULL	171,14
5	34	1	3	PB 95	48,04	5,48	0,05	250,1
6	44	1	2	PB 95	31,18	5,39	0,05	159,66
7	46	1	3	PB 95	27,77	5,39	NULL	149,68
8	49	1	1	PB 95	17,4	5,39	NULL	93,79
9	78	1	2	PB 95	20,75	5,45	0,1	101,78
10	79	1	2	PB 95	30,62	5,45	NULL	166,88

2. Całkowity koszt dostawy

Widok "vw_total_supply_cost" służy do wyliczenia i zaprezentowania całkowitego kosztu każdej dostawy paliwa. Uwzględnia on również takie informacje jak nazwa dostawcy, adres, miasto, kraj oraz telefon kontaktowy. Zawiera także szczegóły dotyczące dostarczanego paliwa, takie jak jego nazwa, cena jednostkowa oraz całkowita ilość dostarczonego paliwa. (FA)

```
CREATE VIEW vw_total_supply_cost AS
SELECT
    s.supply_id,
    s.amount,
    s.date,
    p.name AS petrol_name,
    p.price AS unit_price,
    (s.amount * p.price) AS total_cost,
    supp.company_name,
    supp.address AS supplier_address,
    supp.city AS supplier_city,
    supp.country AS supplier_country,
    supp.phone AS supplier_phone
FROM supply s
JOIN petrol p ON s.petrol_id = p.petrol_id
JOIN supplier supp ON s.supplier_id = supp.supplier_id;
```

	supply_id	amount	date	petrol_name	unit_price	total_cost	company_name	supplier_address	supplier_city	supplier_country	supplier_phone
1	1	4000	2022-06-08 12:00:00.000	PB 95	5,50	22000	Galon	Warszawska 1	Warszawa	Polska	123-456-789
2	2	3000	2022-06-08 12:00:00.000	PB 98	5,70	17100	Galon	Warszawska 1	Warszawa	Polska	123-456-789
3	3	4000	2022-06-08 15:00:00.000	ON	6,20	24800	Azarex	Wrocławska 20	Wrocław	Polska	723-456-789
4	4	3000	2022-06-08 15:00:00.000	ON Plus	6,50	19500	Borim	Poznańska 30	Poznań	Polska	587-654-321
5	5	1200	2022-06-08 12:00:00.000	LPG	2,86	3432	Orlen	Gdańska 40	Gdańsk	Polska	723-456-789
6	6	4000	2022-12-08 12:00:00.000	PB 95	5,50	22000	Galon	Warszawska 1	Warszawa	Polska	123-456-789
7	7	3000	2022-12-08 12:00:00.000	PB 98	5,70	17100	Galon	Warszawska 1	Warszawa	Polska	123-456-789
8	8	4000	2022-12-08 15:00:00.000	ON	6,20	24800	Azarex	Wrocławska 20	Wrocław	Polska	723-456-789
9	9	3000	2022-12-08 15:00:00.000	ON Plus	6,50	19500	Borim	Poznańska 30	Poznań	Polska	587-654-321
10	10	4000	2023-06-08 12:00:00.000	PB 95	5,50	22000	Galon	Warszawska 1	Warszawa	Polska	123-456-789

3. Status dystrybutorów i pistoletów

Widok "vw_distributor_pump_status" służy do prezentowania statusu dystrybutorów oraz przypisanych do nich pistoletów. Uwzględnia on również informacje o rodzaju paliwa oraz cenie. Widok ten może być używany do monitorowania stanu sprzętu. (FA)

```
CREATE VIEW vw_distributor_pump_status AS
SELECT
    d.distributor_no,
    d.status AS distributor_status,
    p.pump_id,
    p.status AS pump_status,
    pet.name AS petrol_name
FROM distributor d
JOIN pump p ON d.distributor_no = p.distributor_no
JOIN petrol pet ON p.petrol_id = pet.petrol_id;
```

	distributor_no	distributor_status	pump_id	pump_status	petrol_name
1	1	down	1	down	PB 95
2	1	down	2	down	PB 98
3	1	down	3	down	ON
4	1	down	4	down	ON Plus
5	1	down	5	down	PB 95
6	1	down	6	down	PB 98
7	1	down	7	down	ON
8	1	down	8	down	ON Plus
9	2	up	9	up	PB 95
10	2	up	10	up	PB 98

4. Szczegóły wszystkich faktur

Widok "vw_invoice_details" służy do prezentacji szczegółów każdej faktury. Uwzględnia takie informacje jak numer faktury, numer transakcji, NIP, numer rejestracyjny pojazdu, data transakcji, nazwa paliwa, końcowy koszt transakcji po uwzględnieniu zniżki oraz identyfikator pracownika obsługującego transakcję. (FA)

```
CREATE VIEW vw_invoice_details AS
SELECT
    i.invoice_id,
    i.transaction_id,
    i.NIP,
    i.plate,
    t.date AS transaction_date,
    p.name AS petrol_name,
    t.amount,
    ph.price,
    ROUND(t.amount * ph.price *
          CASE
              WHEN d.value IS NULL THEN 1
              ELSE (1 - d.value)
          END, 2) AS final_cost,
    t.employee_id
FROM invoice i
JOIN [transaction] t ON i.transaction_id = t.transaction_id
JOIN pump pmp ON t.pump_id = pmp.pump_id
JOIN petrol p ON pmp.petrol_id = p.petrol_id
join petrol_history ph on ph.petrol_id = p.petrol_id
AND ph.date = (
    select MAX(date)
    from petrol_history
    where date <= t.date)
LEFT JOIN discount d ON t.discount_id = d.discount_id;
```

	invoice_id	transaction_id	NIP	plate	transaction_date	petrol_name	amount	price	final_cost	employee_id
1	23	2	4238728573	VHW4749	2022-06-09 23:52:27.000	PB 95	27,24	5,45	148,46	3
2	27	20	2145194999	SFJ8960	2022-06-30 08:42:26.000	PB 95	33,58	5,45	164,71	2
3	42	17	1137573509	ILJ7027	2022-06-25 09:43:48.000	PB 95	23,14	5,45	126,11	1
4	2	31	0670224090	KGM1114	2022-07-14 17:29:10.000	PB 95	31,23	5,48	171,14	2
5	36	29	9075719810	QSK5961	2022-07-11 06:25:03.000	PB 95	16,63	5,48	86,58	1
6	45	52	5546118404	VBH2364	2022-08-23 21:35:28.000	PB 95	41,18	5,39	199,76	2
7	52	44	1513907297	JRT0317	2022-08-10 04:06:46.000	PB 95	31,18	5,39	159,66	2
8	9	64	2172436664	LVJ1765	2022-09-23 12:19:17.000	PB 95	17,78	5,64	100,28	3
9	29	78	6309293305	UVM9306	2022-10-15 15:54:41.000	PB 95	20,75	5,45	101,78	2
10	39	88	7790155229	QWB4025	2022-10-28 20:44:30.000	PB 95	36,99	5,45	201,6	3

5. Całkowita sprzedaż danego paliwa przez lata

Widok vw_total_year_amount został stworzony w celu uzyskania rocznego podsumowania sprzedaży paliwa. Widok zawiera dane dotyczące sumy sprzedanego paliwa oraz łącznej wartości sprzedaży dla każdego rodzaju paliwa, z uwzględnieniem ewentualnych zniżek, w podziale na lata. (FP)

```
create view vw_total_year_amount as
select YEAR(date) AS year, SUM(t.amount) as total_amount,name from [transaction]
as t
left join discount d on d.discount_id = t.discount_id
join pump on pump.pump_id = t.pump_id
join petrol as p on p.petrol_id = pump.petrol_id
group by p.petrol_id,YEAR(date), name
```

	year	total_amount	name
1	2022	1195,23	PB 95
2	2022	789,91	PB 98
3	2022	994,11	ON
4	2022	871,29	ON Plus
5	2022	238,72	LPG
6	2023	1345,82	PB 95
7	2023	1167,28	PB 98
8	2023	1387,95	ON
9	2023	1618,34	ON Plus
10	2023	662,38	LPG

6. Aktywne zniżki

Widok "vw_active_discounts" został stworzony w celu uzyskania listy aktywnych zniżek, które są aktualnie ważne, tzn. takich, których data zakończenia jest pusta (nieokreślona) lub przypada na późniejszy termin niż bieżąca data. (FP)

```
CREATE VIEW vw_active_discounts AS
SELECT discount_id, discount_name, value, start_date, end_date
FROM discount
WHERE end_date IS NULL OR end_date > GETDATE();
```

	discount_id	discount_name	value	start_date	end_date
1	1	Zniżka pracownicza	0,1	2022-06-08 00:00:00.000	2052-06-08 00:00:00.000

7. Średnia cena paliwa w miesiącu

Widok "vw_avg_monthly_petrol_price" powstał do obliczania średniej ceny paliwa dla każdego miesiąca w danym roku, z podziałem na poszczególne rodzaje paliwa.

```
create view vw_avg_monthly_petrol_price as
select p.petrol_id, year(date) as year, month(date) as month, name, avg(ph.price)
as average_price
from dbo.petrol_history as ph
join dbo.petrol p on ph.petrol_id = p.petrol_id
group by p.petrol_id, year(date), month(date), name
```

	petrol_id	year	month	name	average_price
1	1	2022	6	PB 95	5,45
2	1	2022	7	PB 95	5,48
3	1	2022	8	PB 95	5,39
4	1	2022	9	PB 95	5,64
5	1	2022	10	PB 95	5,45
6	1	2022	11	PB 95	5,45
7	1	2022	12	PB 95	5,45
8	1	2023	1	PB 95	5,45
9	1	2023	2	PB 95	5,45
10	1	2023	3	PB 95	5,45

8. Całkowita ilość danego paliwa jaka została zatankowana z danego dystrybutora.

Widok "vw_distributor_fuel_usage" jest zaprojektowany do monitorowania zużycia paliwa przez dystrybutory na stacji paliw. Jego głównym celem jest dostarczenie informacji o całkowitej ilości paliwa sprzedanej przez każdy dystrybutor dla każdego rodzaju paliwa. (FP)

```
create view vw_distributor_fuel_usage as
select d.distributor_no, p.petrol_id,YEAR(date) as year,MONTH(date) as month,
pet.name, SUM(t.amount) as total_amount
from dbo.pump p
join dbo.[transaction] t ON p.pump_id = t.pump_id
join dbo.distributor d ON p.distributor_no = d.distributor_no
join dbo.petrol pet on pet.petrol_id = p.petrol_id
group by d.distributor_no,p.petrol_id, YEAR(date),MONTH(date), pet.name
```

	distributor_no	petrol_id	year	month	name	total_amount
1	1	1	2022	6	PB 95	147,71
2	1	1	2022	7	PB 95	160,76
3	1	1	2022	8	PB 95	106,23
4	1	1	2022	9	PB 95	33,75
5	1	1	2022	10	PB 95	88,36
6	1	1	2022	12	PB 95	216,68
7	1	1	2023	1	PB 95	77,92
8	1	1	2023	2	PB 95	101,27
9	1	1	2023	3	PB 95	64,06
10	1	1	2023	4	PB 95	72,75

Procedury/funkcje

(dla każdej procedury/funkcji należy wkleić kod polecenia definiującego procedurę wraz z komentarzem)

1. Procedura dodawania nowej dostawy paliwa

Procedura sp_add_new_supply została stworzona w celu dodania nowej dostawy paliwa do bazy danych. Procedura wprowadza nowe dane dostawy do tabeli supply, a następnie aktualizuje stan magazynowy odpowiedniego paliwa w tabeli petrol, zwiększając jego ilość o dostarczoną ilość paliwa. W przypadku wystąpienia błędu transakcja jest wycofywana, aby zachować spójność danych. (FP)

```
CREATE PROCEDURE dbo.sp_add_new_supply
    @supplier_id INT,
    @amount FLOAT,
    @date DATETIME,
    @petrol_id INT,
    @price MONEY
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        IF NOT EXISTS (SELECT 1 FROM supplier WHERE supplier_id = @supplier_id)
        BEGIN
            THROW 50005, 'Nie ma dostawcy o takim ID', 1;
        END;

        IF NOT EXISTS (SELECT 1 FROM petrol WHERE petrol_id = @petrol_id)
        BEGIN
            THROW 50005, 'Nie ma paliwa o takim ID', 1;
        END;

        IF @date < (SELECT MAX(date) FROM supply)
        BEGIN
            THROW 50002, 'Data dostawy nie może być wcześniejsza niż ostatnia
dostawa', 1;
        END;

        -- Dodanie nowej dostawy
        INSERT INTO supply (supplier_id, amount, date, petrol_id, price)
```

```
        VALUES (@supplier_id, @amount, @date, @petrol_id, @price);

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0
        BEGIN
            ROLLBACK TRANSACTION;
        END;
        THROW;
    END CATCH;
END;
```

Sposób użycia:

```
exec sp_add_new_supply @supplier_id = 1, @amount = 1500, @date = '2024-06-12',
@petrol_id = 3, @price = 5.16
```

2. Procedura rozpoczęcia procesu tankowania

Procedura sp_start_fueling umożliwia rozpoczęcie tankowania, sprawdzając stan dystrybutora i pistoletu przed zmianą statusu pistoletu na "up". (FP)

```
CREATE PROCEDURE sp_start_fueling
    @pump_id INT
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        -- Sprawdź status dystrybutora dla danego pistoletu
        DECLARE @distributor_status VARCHAR(50);
        SELECT @distributor_status = d.status
        FROM pump p
        JOIN distributor d ON p.distributor_no = d.distributor_no
        WHERE p.pump_id = @pump_id;

        -- Jeżeli status dystrybutora jest 'down', nie można rozpocząć tankowania
        IF @distributor_status = 'down'
        BEGIN
            RAISERROR ('Dystrybutor nieczynny. Nie można rozpocząć tankowania',
16, 1);
            ROLLBACK TRANSACTION;
            RETURN;
        END;

        -- Sprawdzenie, czy pistolet jest w stanie "down" / "up"
        IF EXISTS (
            SELECT 1
            FROM dbo.pump
            WHERE pump_id = @pump_id
            AND status = 'in use'
        )
        BEGIN
            RAISERROR ('Ktoś już tankuje. Proszę wybrać inny dystrybutor.', 16,
1);
            ROLLBACK TRANSACTION;
            RETURN;
        END;

        IF EXISTS (
            SELECT 1
            FROM dbo.pump
            WHERE pump_id = @pump_id
            AND status = 'up'
        )
        BEGIN
            -- Zmiana stanu pistoletu na "in use"
            UPDATE dbo.pump
            SET status = 'in use'
```

```
            WHERE pump_id = @pump_id;

            RAISERROR ('Rozpoczęto tankowanie. Status pistoletu zmieniony na "in
use".', 1, 1);
            COMMIT TRANSACTION;
        END
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION;
        THROW;
    END CATCH;
END;
```

Sposób użycia:

```
exec sp_start_fueling @pump_id = 9
```

3. Procedura zakończenia procesu tankowania

Procedura sp_finish_refueling jest używana do zakończenia procesu tankowania. Obejmuje kilka kroków: sprawdzenie, czy jest wystarczająca ilość paliwa w zapasie, dodanie nowej transakcji, aktualizację zapasu paliwa oraz zmianę statusu pistoletu na "up".

```
CREATE   PROCEDURE sp_finish_refueling
    @pump_id INT,
    @employee_id INT,
    @amount FLOAT,
    @discount_id INT = NULL
AS
BEGIN
    DECLARE @petrol_id INT;
    DECLARE @current_stock FLOAT;

    BEGIN TRY
        BEGIN TRANSACTION;

        -- Odczytaj jakie paliwo jest na danym pistolecie
        SELECT @petrol_id = petrol_id
        FROM pump
        WHERE pump_id = @pump_id;

        -- Odczytaj aktualny zapas paliwa
        SELECT @current_stock = in_stock
        FROM petrol
        WHERE petrol_id = @petrol_id;

        -- Sprawdź czy jest wystarczająco paliwa w zapasie
        IF @current_stock >= @amount
        BEGIN
```

```
            -- Dodaj nową transakcję
            INSERT INTO dbo.[transaction] (pump_id, amount, employee_id, date,
discount_id)
            VALUES (@pump_id, @amount, @employee_id, GETDATE(), @discount_id);

            COMMIT TRANSACTION;
        END
        ELSE
        BEGIN
            -- Jeśli brakuje paliwa
            RAISERROR('Niewystarczająca ilość paliwa.', 16, 1);
            ROLLBACK TRANSACTION;
        END
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0
            ROLLBACK TRANSACTION;
        RAISERROR ('Błąd w użyciu procedury', 16, 1);
    END CATCH
END;
```

Sposób użycia:

```
exec sp_finish_refueling @pump_id = 9, @amount = 44.5, @employee_id = 1
```

4. Procedura aktualizacji statusu dystrybutora

Procedura "sp_update_distributor_status" służy do aktualizowania statusu dystrybutora w bazie danych, a także aktualizowania statusu powiązanych pomp w zależności od nowego statusu dystrybutora. (FP)

```
CREATE PROCEDURE sp_update_distributor_status
    @distributor_no INT,
    @new_status VARCHAR(50)
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        -- Zmieniamy status dystrybutora
        UPDATE dbo.distributor
        SET status = @new_status
        WHERE distributor_no = @distributor_no;

        -- Jeżeli nowy status dystrybutora to 'down', zmieniamy statusy
powiązanych pomp na 'down'
        IF @new_status = 'down'
        BEGIN
            UPDATE dbo.pump
            SET status = 'down'
            WHERE distributor_no = @distributor_no;
        END;

        -- Jeżeli nowy status dystrybutora to 'up', zmieniamy statusy powiązanych
pomp na 'up'
        IF @new_status = 'up'
        BEGIN
            UPDATE dbo.pump
            SET status = 'up'
            WHERE distributor_no = @distributor_no;
        END;

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0
        BEGIN
            ROLLBACK TRANSACTION;
        END;
        THROW;
    END CATCH;
END;
```

Sposób użycia:

```
exec sp_update_distributor_status @distributor_no = 1, @new_status = 'down'
exec sp_update_distributor_status @distributor_no = 1, @new_status = 'up'
```

5. Procedura aktualizacji cen paliwa

Procedura "sp_update_fuel_price" aktualizowania ceny paliwa w bazie danych dla określonego typu paliwa oraz rejestrowania historii zmian cen.

```
CREATE PROCEDURE sp_update_fuel_price
@petrol_id INT,
@new_price MONEY
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        IF NOT EXISTS (SELECT 1 FROM petrol WHERE petrol_id = @petrol_id)
        BEGIN
            THROW 50005, 'Nie ma paliwa o takim ID', 1;
        END;

        -- Aktualizacja ceny paliwa
        UPDATE petrol
        SET price = @new_price
        WHERE petrol_id = @petrol_id;

        -- Dodanie wpisu do historii cen paliwa
        INSERT INTO petrol_history (petrol_id, price, date)
        VALUES (@petrol_id, @new_price, GETDATE());

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        -- Sprawdzenie, czy transakcja jest aktywna i wycofanie jej w przypadku
błędu
        IF @@TRANCOUNT > 0
        BEGIN
            ROLLBACK TRANSACTION;
        END;

        -- Zgłoszenie błędu
        THROW;
    END CATCH;
END;
```

Sposób użycia:

```
exec sp_update_fuel_price @petrol_id = 1, @new_price = 5.50
```

6. Procedura dodanie zniżki z walidacją

Procedura "sp_add_new_discount_with_validation" dodaje nową zniżkę do tabeli discount po sprawdzeniu poprawności danych wejściowych. Procedura waliduje, czy wartość zniżki jest między 0 a 1 oraz czy data rozpoczęcia zniżki jest wcześniejsza niż data zakończenia. Jeśli walidacja przejdzie pomyślnie, nowa zniżka zostaje dodana do tabeli. (FA)

```
CREATE PROCEDURE sp_add_new_discount_with_validation (
@discount_name VARCHAR(50),
@value FLOAT,
@start_date DATETIME,
@end_date DATETIME
)
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        IF @value <= 0 OR @value > 1
        BEGIN
            THROW 50001, 'Zniżka musi należeć do przedziału od 0 do 1', 1;
        END

        IF @start_date >= @end_date
        BEGIN
            THROW 50002, 'Data początkowa nie może być później niż data końcowa',
1;
        END

        INSERT INTO discount (discount_name, value, start_date, end_date)
        VALUES (@discount_name, @value, @start_date, @end_date);

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION;
        THROW;
    END CATCH
END;
```

Sposób użycia:

```
exec sp_add_new_discount_with_validation @discount_name = 'zniżka test', @value =
0.03, @start_date = '2024-06-05', @end_date = '2025-06-05'
```

7. Procedura generująca raport sprzedaży dla danego okresu

Procedura "sp_generate_sales_report" generuje raport sprzedaży dla danego okresu, grupując dane według pracowników i rodzajów paliwa. Wykorzystuje tabele transaction, pump, petrol i employee, aby uzyskać informacje o sprzedaży. Raport zawiera imię i nazwisko pracownika, nazwę paliwa, całkowitą ilość sprzedanego paliwa oraz całkowitą wartość sprzedaży. (FA)

```
CREATE PROCEDURE sp_generate_sales_report (
@start_date DATETIME,
@end_date DATETIME
)
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        IF @start_date >= @end_date
        BEGIN
            THROW 50002, 'Data początkowa nie może być później niż data końcowa',
1;
        END

        SELECT
            e.firstname + ' ' + e.lastname AS employee_name,
            p.name AS petrol_name,
            SUM(t.amount) AS total_amount,
            SUM(t.amount * p.price) AS total_sales
        FROM [transaction] t
        JOIN pump pu ON t.pump_id = pu.pump_id
        JOIN petrol p ON pu.petrol_id = p.petrol_id
        JOIN employee e ON t.employee_id = e.employee_id
        WHERE t.date BETWEEN @start_date AND @end_date
        GROUP BY e.firstname, e.lastname, p.name
        ORDER BY total_sales DESC;

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0
        BEGIN
            ROLLBACK TRANSACTION;
        END;
        THROW;
    END CATCH;
END;
```

Sposób użycia:

```
exec sp_generate_sales_report @start_date = '2023-01-01', @end_date = '2024-01-01'
```

**8. Suma obsłużonych transakcji przez pracownika

Procedura "dbo.sp_employee_transaction_count" do liczenia liczby transakcji obsłużonych przez danego pracownika

```
CREATE PROCEDURE dbo.sp_employee_transaction_count
    @employee_id INT,
    @start_date DATE,
    @end_date DATE
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        IF NOT EXISTS (SELECT 1 FROM employee WHERE employee_id = @employee_id)
        BEGIN
            THROW 50001, 'Nie istnieje pracownik o podanym ID.', 1;
        END;

        SELECT
            employee_id,
            COUNT(transaction_id) AS transaction_count
        FROM
            [transaction] t
        WHERE
            employee_id = @employee_id
            AND t.date >= @start_date
            AND t.date <= @end_date
        GROUP BY
            employee_id;

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0
        BEGIN
            ROLLBACK TRANSACTION;
        END;
        THROW;
    END CATCH;
END;
```

Sposób użycia:

```
exec sp_employee_transaction_count @employee_id = 2, @start_date = '2023-01-01',
@end_date = '2024-01-01'
```

9. Funkcja sprawdzania sprzedaży paliwa w danym okresie

Funkcja "fn_fuel_sales_by_date" służy do uzyskania zestawienia sprzedaży paliwa w określonym przedziale czasowym.

```
CREATE FUNCTION fn_fuel_sales_by_date
(
    @start_date DATETIME,
    @end_date DATETIME
)
RETURNS TABLE
AS
RETURN
(
    SELECT t.transaction_id, t.date, t.amount, p.name AS petrol_name, (t.amount *
p.price) AS total_price
    FROM [transaction] t
    JOIN pump pu ON t.pump_id = pu.pump_id
    JOIN petrol p ON pu.petrol_id = p.petrol_id
    WHERE t.date BETWEEN @start_date AND @end_date
);
GO
```

Sposób użycia:

```
select * from fn_fuel_sales_by_date('2024-01-01', '2024-12-31')
```

10. Funkcja sprawdzania statusu dystrybutora

Funkcja "fn_check_pump_status" służy do sprawdzania statusu pistoletu. (FP)

```
CREATE FUNCTION fn_check_pump_status (@pump_id INT)
RETURNS VARCHAR(50)
AS
BEGIN
    DECLARE @distributor_status VARCHAR(50);

    SELECT @distributor_status = d.status
    FROM pump p
    JOIN distributor d ON p.distributor_no = d.distributor_no
    WHERE p.pump_id = @pump_id;

    RETURN @distributor_status;
END;
GO
```

Sposób użycia:

```
select dbo.fn_check_pump_status(@pump_id)
```

11. Funkcja do sprawdzania zmiany cen paliwa

Funkcja "fn_price_change" oblicza procentową zmianę ceny danego typu paliwa w określonym przedziale czasowym.

```
CREATE FUNCTION fn_price_change (
    @start_date DATETIME,
    @end_date DATETIME,
    @petrol_name VARCHAR(50)
)
RETURNS FLOAT
AS
BEGIN
    DECLARE @start_price FLOAT;
    DECLARE @end_price FLOAT;
    DECLARE @percentage_change FLOAT;

    -- Cena paliwa na początku okresu
    SELECT TOP 1 @start_price = ph.price
    FROM petrol_history ph
    JOIN petrol p ON ph.petrol_id = p.petrol_id
    WHERE p.name = @petrol_name AND ph.date = @start_date
    ORDER BY ph.date DESC;

    -- Cena paliwa na końcu okresu
    SELECT TOP 1 @end_price = ph.price
    FROM petrol_history ph
    JOIN petrol p ON ph.petrol_id = p.petrol_id
    WHERE p.name = @petrol_name AND ph.date = @end_date
    ORDER BY ph.date DESC;

    -- Procentowa zmiana ceny
    IF @start_price IS NULL OR @end_price IS NULL
    BEGIN
        RETURN NULL; -- Zwróć NULL, jeśli nie ma takich dat
    END

    SET @percentage_change = ((@end_price - @start_price) / @start_price) * 100;

    RETURN ROUND(@percentage_change, 1);
END;
GO
```

Sposób użycia:

```
select dbo.fn_price_change('2024-01-01', '2024-06-1', 'PB 95')
```

12. Funkcja do sprawdzania całkowitej wartości dostarczonego paliwa przez dostawcę

Funkcja "fn_total_supply_cost_per_supplier" oblicza łączną wartość dostaw dla określonego dostawcy. (FP)

```
CREATE FUNCTION fn_total_supply_cost_per_supplier (@supplier_id INT)
RETURNS MONEY
AS
BEGIN
    DECLARE @total_cost MONEY;

    SELECT @total_cost = SUM(amount * price)
    FROM supply
    WHERE supplier_id = @supplier_id;

    RETURN @total_cost;
END;
```

Sposób użycia:

```
select dbo.fn_total_supply_cost_per_supplier(2)
```

Triggery

(dla każdego triggera należy wkleić kod polecenia definiującego trigger wraz z komentarzem)

1. Trigger aktualizujący stan magazynowy paliwa po każdej transakcji Trigger "tg_update_petrol_stock_after_transaction" automatycznie aktualizuje ilość paliwa w magazynie po każdej nowej transakcji. Gdy nowa transakcja zostanie dodana do tabeli transaction, ilość paliwa (amount) użytego w tej transakcji zostaje odjęta od dostępnej ilości paliwa (in_stock) w tabeli petrol. (FA)

```
CREATE TRIGGER tg_update_petrol_stock_after_transaction
ON [transaction]
AFTER INSERT
AS
BEGIN
    UPDATE petrol
    SET in_stock = in_stock - i.amount
    FROM inserted i
    JOIN pump p ON p.pump_id = i.pump_id
    WHERE petrol.petrol_id = p.petrol_id;
END;
```

2. Trigger sprawdzający, czy data rozpoczęcia rabatu jest wcześniejsza niż data zakończenia Trigger "tg_check_discount_dates" sprawdza poprawność dat rabatu. Po dodaniu lub aktualizacji rekordu w tabeli discount, trigger weryfikuje, czy data rozpoczęcia (start_date) jest wcześniejsza niż data zakończenia (end_date). Jeśli data rozpoczęcia jest późniejsza lub równa dacie zakończenia, transakcja zostaje cofnięta, a użytkownik otrzymuje komunikat o błędzie. (FA)

```
CREATE TRIGGER tg_check_discount_dates
ON discount
AFTER INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted
        WHERE start_date >= end_date
    )
    BEGIN
        RAISERROR ('Start date must be earlier than end date', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
```

3. Trigger sprawdzający poprawność numeru NIP przed wstawieniem nowej faktury Trigger "tg_validate_invoice_nip" weryfikuje poprawność numeru NIP przed dodaniem nowej faktury do tabeli invoice. Po dodaniu nowej faktury, trigger sprawdza, czy numer NIP ma dokładnie 10 znaków. Jeśli długość numeru NIP jest inna niż 10 znaków, transakcja zostaje cofnięta, a użytkownik otrzymuje komunikat o błędzie. (FA)

```
CREATE TRIGGER tg_validate_invoice_nip
ON invoice
AFTER INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted
        WHERE LEN(NIP) != 10
    )
    BEGIN
        RAISERROR ('Invalid NIP number', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
```

4. Trigger aktualizujący zapas paliwa po pojawieniu się nowej dostawy w tabeli supply Trigger "tg_update_stock" automatycznie aktualizuje zapas paliwa w tabeli petrol po dodaniu nowej dostawy do tabeli supply. Po wstawieniu nowej dostawy, trigger zwiększa wartość zapasu paliwa o ilość dostarczoną, przypisaną do odpowiedniego identyfikatora paliwa (petrol_id).

```
CREATE OR ALTER TRIGGER [dbo].[tg_update_stock]
ON [dbo].[supply]
AFTER INSERT
AS
BEGIN
    UPDATE p
    SET p.in_stock = p.in_stock + i.amount
    FROM petrol p
    JOIN inserted i ON p.petrol_id = i.petrol_id;
END;
```

5.Trigger aktualizujący status pistoletu po pojawieniu się nowej transakcji w tabeli [transaction] Trigger "tg_update_pump_status" aktualizuje status pistoletu w tabeli pump po pojawieniu się nowej transakcji w tabeli [transaction]. Po dodaniu nowej transakcji, trigger ustawia status odpowiedniego pistoletu na "up", przypisując go do identyfikatora pistoletu (pump_id).

```
CREATE OR ALTER TRIGGER [dbo].[tg_update_pump_status]
ON [dbo].[transaction]
AFTER INSERT
AS
BEGIN
    UPDATE pump
    SET status = 'up'
    FROM pump p
    JOIN inserted AS i
```

```
    ON p.pump_id = i.pump_id;
END;
```