-- Table: public.Garanti

-- DROP TABLE IF EXISTS public."Garanti";

CREATE TABLE IF NOT EXISTS public."Garanti"

(

"GarantiID" integer NOT NULL DEFAULT nextval('"Garanti\_GarantiID\_seq"'::regclass),

"ÜrünID" integer NOT NULL,

"BaşlangıçTarihi" date NOT NULL,

"BitişTarihi" date NOT NULL,

"GarantiDurumu" character varying(50) COLLATE pg\_catalog."default" DEFAULT 'Geçerli'::character varying,

CONSTRAINT "Garanti\_pkey" PRIMARY KEY ("GarantiID"),

CONSTRAINT "Garanti\_ÜrünID\_fkey" FOREIGN KEY ("ÜrünID")

REFERENCES public."Ürünler" ("ÜrünID") MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Garanti"

OWNER to postgres;

-- Table: public.Kategoriler

-- DROP TABLE IF EXISTS public."Kategoriler";

CREATE TABLE IF NOT EXISTS public."Kategoriler"

(

"KategoriID" integer NOT NULL DEFAULT nextval('"Kategoriler\_KategoriID\_seq"'::regclass),

"KategoriAdı" character varying(255) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Kategoriler\_pkey" PRIMARY KEY ("KategoriID")

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Kategoriler"

OWNER to postgres;

-- Table: public.Kişi

-- DROP TABLE IF EXISTS public."Kişi";

CREATE TABLE IF NOT EXISTS public."Kişi"

(

"KişiID" integer NOT NULL DEFAULT nextval('"Kişi\_KişiID\_seq"'::regclass),

"Ad" character varying(50) COLLATE pg\_catalog."default" NOT NULL,

"Soyad" character varying(50) COLLATE pg\_catalog."default" NOT NULL,

"Email" character varying(100) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Kişi\_pkey" PRIMARY KEY ("KişiID")

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Kişi"

OWNER to postgres;

-- Table: public.MüşteriAdresleri

-- DROP TABLE IF EXISTS public."MüşteriAdresleri";

CREATE TABLE IF NOT EXISTS public."MüşteriAdresleri"

(

"AdresID" integer NOT NULL DEFAULT nextval('"MüşteriAdresleri\_AdresID\_seq"'::regclass),

"MüşteriID" integer NOT NULL,

"Adres" character varying(255) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "MüşteriAdresleri\_pkey" PRIMARY KEY ("AdresID"),

CONSTRAINT "MüşteriAdresleri\_MüşteriID\_fkey" FOREIGN KEY ("MüşteriID")

REFERENCES public."Müşteriler" ("MüşteriID") MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."MüşteriAdresleri"

OWNER to postgres;

-- Table: public.Müşteriler

-- DROP TABLE IF EXISTS public."Müşteriler";

CREATE TABLE IF NOT EXISTS public."Müşteriler"

(

"MüşteriID" integer NOT NULL DEFAULT nextval('"Müşteriler\_MüşteriID\_seq"'::regclass),

"KişiID" integer NOT NULL,

"Telefon" character varying(20) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Müşteriler\_pkey" PRIMARY KEY ("MüşteriID"),

CONSTRAINT "Müşteriler\_KişiID\_fkey" FOREIGN KEY ("KişiID")

REFERENCES public."Kişi" ("KişiID") MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Müşteriler"

OWNER to postgres;

-- Table: public.Satıcılar

-- DROP TABLE IF EXISTS public."Satıcılar";

CREATE TABLE IF NOT EXISTS public."Satıcılar"

(

"SatıcıID" integer NOT NULL DEFAULT nextval('"Satıcılar\_SatıcıID\_seq"'::regclass),

"Ad" character varying(255) COLLATE pg\_catalog."default" NOT NULL,

"Soyad" character varying(255) COLLATE pg\_catalog."default" NOT NULL,

"MağazaAdı" character varying(255) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Satıcılar\_pkey" PRIMARY KEY ("SatıcıID")

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Satıcılar"

OWNER to postgres;

-- Table: public.Satışlar

-- DROP TABLE IF EXISTS public."Satışlar";

CREATE TABLE IF NOT EXISTS public."Satışlar"

(

"SatışID" integer NOT NULL DEFAULT nextval('"Satışlar\_SatışID\_seq"'::regclass),

"ÇalışanID" integer NOT NULL,

"SiparişID" integer NOT NULL,

"Tarih" date NOT NULL,

"ToplamTutar" numeric(10,2),

CONSTRAINT "Satışlar\_pkey" PRIMARY KEY ("SatışID"),

CONSTRAINT "Satışlar\_SiparişID\_fkey" FOREIGN KEY ("SiparişID")

REFERENCES public."Siparişler" ("SiparişID") MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE,

CONSTRAINT "Satışlar\_ÇalışanID\_fkey" FOREIGN KEY ("ÇalışanID")

REFERENCES public."Çalışanlar" ("ÇalışanID") MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

-- Table: public.SiparişDetayları

-- DROP TABLE IF EXISTS public."SiparişDetayları";

CREATE TABLE IF NOT EXISTS public."SiparişDetayları"

(

"DetayID" integer NOT NULL DEFAULT nextval('"SiparişDetayları\_DetayID\_seq"'::regclass),

"SiparişID" integer NOT NULL,

"ÜrünID" integer NOT NULL,

"Miktar" integer NOT NULL,

"BirimFiyat" numeric(10,2) NOT NULL,

CONSTRAINT "SiparişDetayları\_pkey" PRIMARY KEY ("DetayID"),

CONSTRAINT "SiparişDetayları\_SiparişID\_fkey" FOREIGN KEY ("SiparişID")

REFERENCES public."Siparişler" ("SiparişID") MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE,

CONSTRAINT "SiparişDetayları\_ÜrünID\_fkey" FOREIGN KEY ("ÜrünID")

REFERENCES public."Ürünler" ("ÜrünID") MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

-- Table: public.Siparişler

-- DROP TABLE IF EXISTS public."Siparişler";

CREATE TABLE IF NOT EXISTS public."Siparişler"

(

"SiparişID" integer NOT NULL DEFAULT nextval('"Siparişler\_SiparişID\_seq"'::regclass),

"MüşteriID" integer NOT NULL,

"Tarih" date NOT NULL,

"ToplamTutar" numeric(10,2),

"Durum" character varying(50) COLLATE pg\_catalog."default",

CONSTRAINT "Siparişler\_pkey" PRIMARY KEY ("SiparişID"),

CONSTRAINT "Siparişler\_MüşteriID\_fkey" FOREIGN KEY ("MüşteriID")

REFERENCES public."Müşteriler" ("MüşteriID") MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Siparişler"

OWNER to postgres;

-- Table: public.StokLog

-- DROP TABLE IF EXISTS public."StokLog";

CREATE TABLE IF NOT EXISTS public."StokLog"

(

"LogID" integer NOT NULL DEFAULT nextval('"StokLog\_LogID\_seq"'::regclass),

"ÜrünID" integer NOT NULL,

"Tarih" timestamp without time zone NOT NULL DEFAULT now(),

"Mesaj" text COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "StokLog\_pkey" PRIMARY KEY ("LogID"),

CONSTRAINT "StokLog\_ÜrünID\_fkey" FOREIGN KEY ("ÜrünID")

REFERENCES public."Ürünler" ("ÜrünID") MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."StokLog"

OWNER to postgres;

-- Table: public.Yorumlar

-- DROP TABLE IF EXISTS public."Yorumlar";

CREATE TABLE IF NOT EXISTS public."Yorumlar"

(

"YorumID" integer NOT NULL DEFAULT nextval('"Yorumlar\_YorumID\_seq"'::regclass),

"ÜrünID" integer NOT NULL,

"MüşteriID" integer NOT NULL,

"YorumMetni" text COLLATE pg\_catalog."default",

"Puan" integer,

CONSTRAINT "Yorumlar\_pkey" PRIMARY KEY ("YorumID"),

CONSTRAINT "Yorumlar\_MüşteriID\_fkey" FOREIGN KEY ("MüşteriID")

REFERENCES public."Müşteriler" ("MüşteriID") MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE,

CONSTRAINT "Yorumlar\_ÜrünID\_fkey" FOREIGN KEY ("ÜrünID")

REFERENCES public."Ürünler" ("ÜrünID") MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE,

CONSTRAINT "Yorumlar\_Puan\_check" CHECK ("Puan" >= 1 AND "Puan" <= 5)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Yorumlar"

OWNER to postgres;

-- Table: public.Çalışanlar

-- DROP TABLE IF EXISTS public."Çalışanlar";

CREATE TABLE IF NOT EXISTS public."Çalışanlar"

(

"ÇalışanID" integer NOT NULL DEFAULT nextval('"Çalışanlar\_ÇalışanID\_seq"'::regclass),

"KişiID" integer NOT NULL,

"Pozisyon" character varying(50) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Çalışanlar\_pkey" PRIMARY KEY ("ÇalışanID"),

CONSTRAINT "Çalışanlar\_KişiID\_fkey" FOREIGN KEY ("KişiID")

REFERENCES public."Kişi" ("KişiID") MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Çalışanlar"

OWNER to postgres;

-- Table: public.ÖdemeYöntemleri

-- DROP TABLE IF EXISTS public."ÖdemeYöntemleri";

CREATE TABLE IF NOT EXISTS public."ÖdemeYöntemleri"

(

"ÖdemeID" integer NOT NULL DEFAULT nextval('"ÖdemeYöntemleri\_ÖdemeID\_seq"'::regclass),

"MüşteriID" integer NOT NULL,

"Tür" character varying(50) COLLATE pg\_catalog."default" NOT NULL,

"KartNumarası" character varying(20) COLLATE pg\_catalog."default",

CONSTRAINT "ÖdemeYöntemleri\_pkey" PRIMARY KEY ("ÖdemeID"),

CONSTRAINT "ÖdemeYöntemleri\_MüşteriID\_fkey" FOREIGN KEY ("MüşteriID")

REFERENCES public."Müşteriler" ("MüşteriID") MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."ÖdemeYöntemleri"

OWNER to postgres;

-- Table: public.Ürünler

-- DROP TABLE IF EXISTS public."Ürünler";

CREATE TABLE IF NOT EXISTS public."Ürünler"

(

"ÜrünID" integer NOT NULL DEFAULT nextval('"Ürünler\_ÜrünID\_seq"'::regclass),

"ÜrünAdı" character varying(255) COLLATE pg\_catalog."default" NOT NULL,

"KategoriID" integer NOT NULL,

"Fiyat" numeric(10,2) NOT NULL,

"StokMiktarı" integer NOT NULL,

CONSTRAINT "Ürünler\_pkey" PRIMARY KEY ("ÜrünID"),

CONSTRAINT "Ürünler\_KategoriID\_fkey" FOREIGN KEY ("KategoriID")

REFERENCES public."Kategoriler" ("KategoriID") MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."Ürünler"

OWNER to postgres;

-- Table: public.İadeTalepleri

-- DROP TABLE IF EXISTS public."İadeTalepleri";

CREATE TABLE IF NOT EXISTS public."İadeTalepleri"

(

"İadeID" integer NOT NULL DEFAULT nextval('"İadeTalepleri\_İadeID\_seq"'::regclass),

"SiparişID" integer NOT NULL,

"Tarih" date NOT NULL,

"Durum" character varying(50) COLLATE pg\_catalog."default" DEFAULT 'İnceleniyor'::character varying,

CONSTRAINT "İadeTalepleri\_pkey" PRIMARY KEY ("İadeID"),

CONSTRAINT "İadeTalepleri\_SiparişID\_fkey" FOREIGN KEY ("SiparişID")

REFERENCES public."Siparişler" ("SiparişID") MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."İadeTalepleri"

OWNER to postgres;

-- Table: public.İndirimler

-- DROP TABLE IF EXISTS public."İndirimler";

CREATE TABLE IF NOT EXISTS public."İndirimler"

(

"İndirimID" integer NOT NULL DEFAULT nextval('"İndirimler\_İndirimID\_seq"'::regclass),

"ÜrünID" integer NOT NULL,

"İndirimOranı" numeric(5,2) NOT NULL,

"BaşlangıçTarihi" date NOT NULL,

"BitişTarihi" date NOT NULL,

CONSTRAINT "İndirimler\_pkey" PRIMARY KEY ("İndirimID"),

CONSTRAINT "İndirimler\_ÜrünID\_fkey" FOREIGN KEY ("ÜrünID")

REFERENCES public."Ürünler" ("ÜrünID") MATCH SIMPLE

ON UPDATE CASCADE

ON DELETE CASCADE,

CONSTRAINT "İndirimler\_İndirimOranı\_check" CHECK ("İndirimOranı" >= 0::numeric AND "İndirimOranı" <= 100::numeric)

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."İndirimler"

OWNER to postgres;

-- Table: public.İstatistikler

-- DROP TABLE IF EXISTS public."İstatistikler";

CREATE TABLE IF NOT EXISTS public."İstatistikler"

(

"İstatistikID" integer NOT NULL DEFAULT nextval('"İstatistikler\_İstatistikID\_seq"'::regclass),

"Tarih" date NOT NULL,

"SatışAdedi" integer NOT NULL DEFAULT 0,

"Gelir" numeric(10,2) NOT NULL DEFAULT 0,

CONSTRAINT "İstatistikler\_pkey" PRIMARY KEY ("İstatistikID")

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public."İstatistikler"

OWNER to postgres

FONKSİYONLAR:

-- FUNCTION: public.calculate\_order\_total(integer)

-- DROP FUNCTION IF EXISTS public.calculate\_order\_total(integer);

CREATE OR REPLACE FUNCTION public.calculate\_order\_total(

order\_id integer)

RETURNS numeric

LANGUAGE 'plpgsql'

COST 100

VOLATILE PARALLEL UNSAFE

AS $BODY$

DECLARE

total\_order\_amount NUMERIC(10, 2);

BEGIN

-- Sipariş toplamını hesapla

SELECT SUM("Miktar" \* "BirimFiyat") INTO total\_order\_amount

FROM public."SiparişDetayları"

WHERE "SiparişID" = order\_id;

-- Eğer sipariş detayları yoksa 0 döndür

RETURN COALESCE(total\_order\_amount, 0);

END;

$BODY$;

ALTER FUNCTION public.calculate\_order\_total(integer)

OWNER TO postgres;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- FUNCTION: public.encoksatanurunler()

-- DROP FUNCTION IF EXISTS public.encoksatanurunler();

CREATE OR REPLACE FUNCTION public.encoksatanurunler(

)

RETURNS TABLE(urunadi character varying, satilanmiktar integer)

LANGUAGE 'plpgsql'

COST 100

VOLATILE PARALLEL UNSAFE

ROWS 1000

AS $BODY$

BEGIN

RETURN QUERY

SELECT

u."ÜrünAdı",

CAST(SUM(sd."Miktar") AS INT) AS SatilanMiktar

FROM "SiparişDetayları" sd

JOIN "Ürünler" u ON sd."ÜrünID" = u."ÜrünID"

GROUP BY u."ÜrünAdı"

ORDER BY SatilanMiktar DESC;

END;

$BODY$;

ALTER FUNCTION public.encoksatanurunler()

OWNER TO postgres;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- FUNCTION: public.get\_all\_categories\_with\_products()

-- DROP FUNCTION IF EXISTS public.get\_all\_categories\_with\_products();

CREATE OR REPLACE FUNCTION public.get\_all\_categories\_with\_products(

)

RETURNS TABLE(categoryname character varying, productname character varying, price numeric)

LANGUAGE 'plpgsql'

COST 100

VOLATILE PARALLEL UNSAFE

ROWS 1000

AS $BODY$

BEGIN

RETURN QUERY

SELECT

k."KategoriAdı", -- Kategori adı (varsayıyorum ki Kategoriler tablosunda bu sütun var)

u."ÜrünAdı",

u."Fiyat"

FROM

public."Ürünler" u

JOIN

public."Kategoriler" k ON u."KategoriID" = k."KategoriID"

ORDER BY

k."KategoriAdı", u."ÜrünAdı"; -- Kategorilere göre sıralama

END;

$BODY$;

ALTER FUNCTION public.get\_all\_categories\_with\_products()

OWNER TO postgres;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- FUNCTION: public.get\_order\_details(integer)

-- DROP FUNCTION IF EXISTS public.get\_order\_details(integer);

CREATE OR REPLACE FUNCTION public.get\_order\_details(

order\_id integer)

RETURNS TABLE(productname character varying, quantity integer, unitprice numeric)

LANGUAGE 'plpgsql'

COST 100

VOLATILE PARALLEL UNSAFE

ROWS 1000

AS $BODY$

BEGIN

RETURN QUERY

SELECT

u."ÜrünAdı" AS ProductName,

sd."Miktar" AS Quantity,

sd."BirimFiyat" AS UnitPrice

FROM

public."SiparişDetayları" sd

JOIN

public."Ürünler" u ON sd."ÜrünID" = u."ÜrünID"

WHERE

sd."SiparişID" = order\_id;

END;

$BODY$;

ALTER FUNCTION public.get\_order\_details(integer)

OWNER TO postgres;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- FUNCTION: public.get\_products\_by\_category(integer)

-- DROP FUNCTION IF EXISTS public.get\_products\_by\_category(integer);

CREATE OR REPLACE FUNCTION public.get\_products\_by\_category(

category\_id integer)

RETURNS TABLE(productname character varying, price numeric)

LANGUAGE 'plpgsql'

COST 100

VOLATILE PARALLEL UNSAFE

ROWS 1000

AS $BODY$

BEGIN

RETURN QUERY

SELECT "ÜrünAdı", "Fiyat"

FROM public."Ürünler"

WHERE "KategoriID" = category\_id;

END;

$BODY$;

ALTER FUNCTION public.get\_products\_by\_category(integer)

OWNER TO postgres;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- FUNCTION: public.get\_top\_selling\_products(date, date)

-- DROP FUNCTION IF EXISTS public.get\_top\_selling\_products(date, date);

CREATE OR REPLACE FUNCTION public.get\_top\_selling\_products(

start\_date date,

end\_date date)

RETURNS TABLE(productname character varying, totalquantitysold bigint)

LANGUAGE 'plpgsql'

COST 100

VOLATILE PARALLEL UNSAFE

ROWS 1000

AS $BODY$

BEGIN

RETURN QUERY

SELECT

u."ÜrünAdı" AS ProductName,

SUM(sd."Miktar") AS TotalQuantitySold

FROM

public."SiparişDetayları" sd

JOIN

public."Ürünler" u ON sd."ÜrünID" = u."ÜrünID"

JOIN

public."Siparişler" s ON sd."SiparişID" = s."SiparişID"

WHERE

s."Tarih" BETWEEN start\_date AND end\_date

GROUP BY

u."ÜrünAdı"

ORDER BY

TotalQuantitySold DESC

LIMIT 10;

END;

$BODY$;

ALTER FUNCTION public.get\_top\_selling\_products(date, date)

OWNER TO postgres;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- FUNCTION: public.mostsoldproduct()

-- DROP FUNCTION IF EXISTS public.mostsoldproduct();

CREATE OR REPLACE FUNCTION public.mostsoldproduct(

)

RETURNS TABLE(productid integer, productname character varying, totalquantity integer)

LANGUAGE 'plpgsql'

COST 100

VOLATILE PARALLEL UNSAFE

ROWS 1000

AS $BODY$

BEGIN

RETURN QUERY

SELECT "ÜrünID", "ÜrünAdı", SUM("Miktar") AS TotalQuantity

FROM "Satışlar"

JOIN "SiparişDetayları" ON "Satışlar"."SiparişID" = "SiparişDetayları"."SiparişID"

JOIN "Ürünler" ON "SiparişDetayları"."ÜrünID" = "Ürünler"."ÜrünID"

GROUP BY "ÜrünID", "ÜrünAdı"

ORDER BY TotalQuantity DESC

LIMIT 1;

END;

$BODY$;

ALTER FUNCTION public.mostsoldproduct()

OWNER TO postgres;

PROSEDÜRLER:

-- PROCEDURE: public.averageproductprice()

-- DROP PROCEDURE IF EXISTS public.averageproductprice();

CREATE OR REPLACE PROCEDURE public.averageproductprice(

)

LANGUAGE 'plpgsql'

AS $BODY$

DECLARE

avg\_price NUMERIC;

BEGIN

-- Tüm ürünlerin ortalama fiyatını hesapla

SELECT AVG("Fiyat") INTO avg\_price

FROM "Ürünler";

-- Hesaplanan ortalama fiyat ile ilgili bir işlem yapılabilir (güncelleme vb.)

-- Bu sadece hesaplama yapar, işlemi buna göre değiştirebilirsiniz.

UPDATE "İstatistikler"

SET "OrtalamaÜrünFiyatı" = avg\_price

WHERE "İstatistikID" = 1;

END;

$BODY$;

ALTER PROCEDURE public.averageproductprice()

OWNER TO postgres;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- PROCEDURE: public.customertotalorderamount(integer)

-- DROP PROCEDURE IF EXISTS public.customertotalorderamount(integer);

CREATE OR REPLACE PROCEDURE public.customertotalorderamount(

IN p\_customerid integer)

LANGUAGE 'plpgsql'

AS $BODY$

DECLARE

total\_amount NUMERIC;

BEGIN

-- Müşteriye ait tüm siparişlerin toplam tutarını hesapla

SELECT SUM("ToplamTutar") INTO total\_amount

FROM "Siparişler"

WHERE "MüşteriID" = p\_CustomerID;

-- Hesaplanan toplam sipariş tutarını güncelle veya başka bir işlem yap

-- Bu işlemi güncelleme olarak düşünebilirsiniz, veya sadece işlemi yapıp döndürmeyebilirsiniz

UPDATE "Müşteriler"

SET "ToplamSiparişTutari" = total\_amount

WHERE "MüşteriID" = p\_CustomerID;

END;

$BODY$;

ALTER PROCEDURE public.customertotalorderamount(integer)

OWNER TO postgres;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- PROCEDURE: public.topsalesamount(date, date)

-- DROP PROCEDURE IF EXISTS public.topsalesamount(date, date);

CREATE OR REPLACE PROCEDURE public.topsalesamount(

IN p\_startdate date,

IN p\_enddate date)

LANGUAGE 'plpgsql'

AS $BODY$

BEGIN

-- Belirtilen tarih aralığındaki toplam satış tutarını hesapla

UPDATE "Satışlar"

SET "ToplamTutar" = (

SELECT SUM("ToplamTutar") FROM "Satışlar"

WHERE "Tarih" BETWEEN p\_StartDate AND p\_EndDate

);

END;

$BODY$;

ALTER PROCEDURE public.topsalesamount(date, date)

OWNER TO postgres;

TRİGGER:

-- FUNCTION: public.kontrol\_dusuk\_puan()

-- DROP FUNCTION IF EXISTS public.kontrol\_dusuk\_puan();

CREATE OR REPLACE FUNCTION public.kontrol\_dusuk\_puan()

RETURNS trigger

LANGUAGE 'plpgsql'

COST 100

VOLATILE NOT LEAKPROOF

AS $BODY$

BEGIN

-- Eğer puan 1 ise uyarı ver

IF NEW."Puan" = 1 THEN

RAISE NOTICE 'Düşük puanlı bir yorum eklendi: ÜrünID=%, YorumID=%', NEW."ÜrünID", NEW."YorumID";

END IF;

RETURN NEW;

END;

$BODY$;

ALTER FUNCTION public.kontrol\_dusuk\_puan()

OWNER TO postgres;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- FUNCTION: public.log\_stock\_zero()

-- DROP FUNCTION IF EXISTS public.log\_stock\_zero();

CREATE OR REPLACE FUNCTION public.log\_stock\_zero()

RETURNS trigger

LANGUAGE 'plpgsql'

COST 100

VOLATILE NOT LEAKPROOF

AS $BODY$

BEGIN

-- Eğer stok miktarı 0 ise log kaydı oluştur

IF NEW."StokMiktarı" = 0 THEN

INSERT INTO public."StokLog" ("ÜrünID", "Mesaj", "Tarih")

VALUES (NEW."ÜrünID", 'Stok miktarı sıfıra düştü.', CURRENT\_TIMESTAMP);

END IF;

RETURN NEW;

END;

$BODY$;

ALTER FUNCTION public.log\_stock\_zero()

OWNER TO postgres;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- FUNCTION: public.update\_monthly\_statistics()

-- DROP FUNCTION IF EXISTS public.update\_monthly\_statistics();

CREATE OR REPLACE FUNCTION public.update\_monthly\_statistics()

RETURNS trigger

LANGUAGE 'plpgsql'

COST 100

VOLATILE NOT LEAKPROOF

AS $BODY$

BEGIN

-- Eğer ilgili ay için bir kayıt varsa, güncelle

UPDATE public."İstatistikler"

SET

"SatışAdedi" = "SatışAdedi" + 1, -- Satış adedini 1 arttır

"Gelir" = "Gelir" + NEW."ToplamTutar" -- Satış gelirini arttır

WHERE "Tarih" = DATE\_TRUNC('month', NEW."Tarih")::date; -- Ay bazında

-- Eğer o ay için bir kayıt yoksa, yeni bir kayıt oluştur

IF NOT FOUND THEN

INSERT INTO public."İstatistikler" ("Tarih", "SatışAdedi", "Gelir")

VALUES (DATE\_TRUNC('month', NEW."Tarih")::date, 1, NEW."ToplamTutar"); -- Yeni kayıt ekle

END IF;

RETURN NEW;

END;

$BODY$;

ALTER FUNCTION public.update\_monthly\_statistics()

OWNER TO postgres;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- FUNCTION: public.update\_sipariş\_durumu\_to\_iade()

-- DROP FUNCTION IF EXISTS public."update\_sipariş\_durumu\_to\_iade"();

CREATE OR REPLACE FUNCTION public."update\_sipariş\_durumu\_to\_iade"()

RETURNS trigger

LANGUAGE 'plpgsql'

COST 100

VOLATILE NOT LEAKPROOF

AS $BODY$

BEGIN

-- İlgili siparişin durumunu "İade" olarak güncelle

UPDATE public."Siparişler"

SET "Durum" = 'İade'

WHERE "SiparişID" = NEW."SiparişID";

RETURN NEW;

END;

$BODY$;

ALTER FUNCTION public."update\_sipariş\_durumu\_to\_iade"()

OWNER TO postgres;