ZADÁNÍ PROJEKTU:

DATABÁZE PRO SPRÁVU ZÁKAZNICKÝCH OBJEDNÁVEK

1. Popis projektu:

Vytvoř databázi pro fiktivní e-shop, který spravuje objednávky zákazníků. Databáze bude obsahovat tabulky pro zákazníky, produkty, objednávky a detail objednávek. Na základě této databáze vytvoříš dotazy a reporty s použitím různých SQL funkcí.

2. Struktura databáze:

Tabulky:

- 1) customers
 - customer_id (PK)
 - first_name
 - last_name
 - email
 - registration_date
 - country
- 2) products
 - product_id (PK)
 - product_name
 - category
 - price (DECIMAL)
 - stock_quantity
- 3) orders
 - order_id (PK)
 - customer_id (FK)
 - order_date
 - total_price (DECIMAL)
 - status
- 4) order_details
 - order_detail_id (PK)
 - order_id (FK)
 - product_id (FK)
 - quantity
 - price_per_unit (DECIMAL)

3. <u>Úkoly k vypracování</u>

- 1) Vytvoření databáze a tabulek dle výše uvedené struktury
- 2) Vložení testovacích dat:
 - 5 zákazníků
 - 5 produktů
 - 3 objednávky, z nichž každá má 2-3 položky

Řešení:

Databázi jsem vytvořila v programu PostgreSQL
 Vytvoření tabulek proběhlo pomocí těchto SQL příkazů:

```
CREATE TABLE customers(
       customer_id SERIAL PRIMARY KEY,
       first_name VARCHAR(30) NOT NULL,
       last_name VARCHAR(30) NOT NULL,
       email VARCHAR(40) UNIQUE NOT NULL,
       registration_date DATE,
       country VARCHAR(50)
);
CREATE TABLE products(
       product_id SERIAL PRIMARY KEY,
       product_name VARCHAR(50) UNIQUE NOT NULL,
       category VARCHAR(50) NOT NULL,
       price NUMERIC(10,2),
       stock_quantity INT
);
CREATE TABLE orders(
       order_id SERIAL PRIMARY KEY,
       customer_id INT REFERENCES customers(customer_id),
       order_date DATE,
       total_price NUMERIC (10,2),
       status VARCHAR(30)
)
CREATE TABLE order_details(
       order_detail_id SERIAL PRIMARY KEY,
       order_id INT REFERENCES orders(order_id),
       product_id INT REFERENCES products(product_id),
       quantity INT,
       price_per_unit NUMERIC(10,2)
)
```

```
Vložení testovacích dat – 5 zákazníků
INSERT INTO customers (first name, last name, email, registration date, country)
VALUES
('Anežka', 'Kohoutová', 'ak@gmail.com', '2024-10-01', 'Czech Republic'),
('Hynek', 'Beran', 'hb@seznam.cz', '2024-10-02', 'Slovakia'),
('Barbora', 'Zemanová', 'bz@seznam.cz', '2024-10-02', 'Czech Republic'),
('Jan', 'Ježek', 'jj@seznam.cz', '2024-10-03', 'Slovakia'),
('Evžen', 'Konvička', 'ek@gmail.com', '2024-10-05', 'Czech Republic');
Kontrola vložení: SELECT * FROM customers;
Vložení testovacích dat – 5 produktů
INSERT INTO products (product_name, category, price, stock_quantity)
VALUES
('Svět zvířat', 'encyklopedie', 339, 50),
('Velká kniha pohádek', 'pohádky', 357, 30),
('Šikmý kostel', 'beletrie', 379, 100),
('Matematika pro gymnázia', 'učebnice', 169, 90),
('Kronika zániku Evropy', 'beletrie', 599, 85);
Kontrola vložení: SELECT * FROM products;
Vložení 3 objednávek, z nichž každá má 2-3 položky
INSERT INTO orders(customer id, order date, total price, status)
VALUES
(1, '2024-10-01', 696, 'vyřízeno'),
(2, '2024-10-02', 978, 'vyřízeno'),
(3, '2024-10-02', 1147, 'vyřízeno'),
(4, '2024-10-03', 1075, 'vyřízeno'),
(5, '2024-10-05', 937, 'vyřízeno');
Kontrola vložení: SELECT * FROM orders;
INSERT INTO order_details(order_id, product_id, quantity, price_per_unit)
VALUES
```

(1, 11, 1, 339), (1, 12, 1, 357), (2, 13, 1, 379), (2, 15, 1, 599), (3, 13, 1, 379), (3, 14, 1, 169),

```
(3, 15, 1, 599),
```

(4, 11, 1, 339),

(4, 12, 1, 357),

(4, 13, 1, 379),

(5, 11, 1, 339),

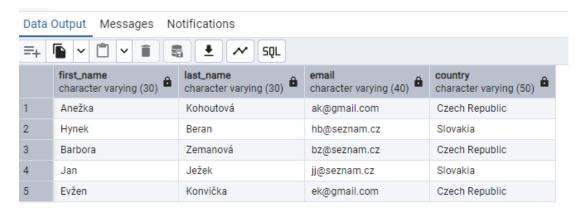
(5, 15, 1, 599);

Kontrola vložení: SELECT * FROM order_details;

4. SQL dotazy k vypracování

a) Vyber seznam všech zákazníků s jejich emaily a zemí

SELECT first_name, last_name, email, country FROM customers;



b) Vyber všechny produkty v kategorii "Beleterie", které jsou skladem

SELECT * FROM products

WHERE category = 'beletrie' AND stock_quantity>1;

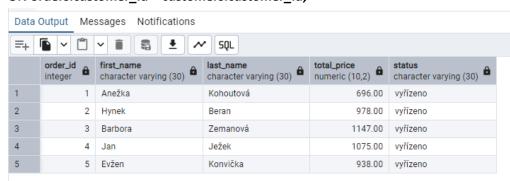


c) Zobraz seznam všech objednávek včetně jména zákazníka, celkové ceny a stavu objednávky

SELECT order_id, customers.first_name, customers.last_name, total_price, status FROM orders

INNER JOIN customers

ON orders.customer_id = customers.customer_id;



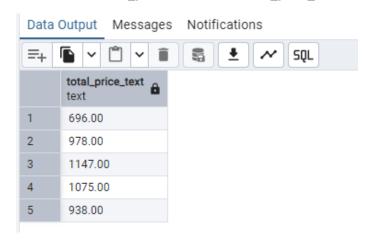
d) Zobraz seznam objednaných produktů včetně názvu produktu, množství a celkové ceny za daný produkt v objednávce

SELECT order_details.order_id, products.product_name, order_details.quantity,
(order_details.quantity * order_details.price_per_unit) AS total_price
FROM order_details
INNER JOIN products
ON order_details.product_id=products.product_id;

Data Output Messages Notifications				
=+ □ ∨ □ ∨ ■ ■ ± ~ SQL				
	order_id integer	product_name character varying (50)	quantity integer	total_price numeric
1	1	Svět zvířat	1	339.00
2	1	Velká kniha pohádek	1	357.00
3	2	Šikmý kostel	1	379.00
4	2	Kronika zániku Evropy	1	599.00
5	3	Šikmý kostel	1	379.00
6	3	Matematika pro gymnázia	1	169.00
7	3	Kronika zániku Evropy	1	599.00
8	4	Svět zvířat	1	339.00
9	4	Velká kniha pohádek	1	357.00
10	4	Šikmý kostel	1	379.00
11	5	Svět zvířat	1	339.00
12	5	Kronika zániku Evropy	1	599.00

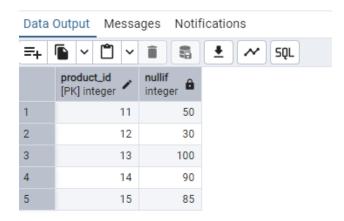
e) Použij funkci CAST ke změně datového typu celkové ceny objednávky na text

SELECT CAST (total_price as TEXT) AS total_price_text FROM orders;



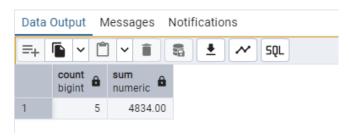
f) Použij funkci NULLIF k vypočítání ceny za jednotku, přičemž pokud je množství 0, vrátí NULL

SELECT product_id, NULLIF (stock_quantity, 0) FROM products;



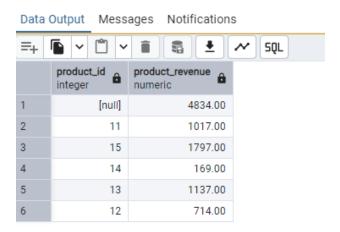
g) Zobraz celkový počet objednávek a jejich hodnotu podle stavu objednávky s použitím GROUP BY

SELECT COUNT(order_id), SUM(total_price) FROM orders GROUP BY status;



h) Použij ROLLUP k vytvoření přehledu tržeb za každý produkt a celkovou tržbu za všechny produkty

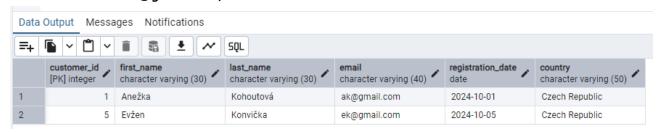
SELECT product_id, SUM(quantity*price_per_unit) AS product_revenue FROM order_details GROUP BY ROLLUP (product_id);



i) Najdi všechny zákazníky, kteří mají email končící na "gmail.com"

SELECT * FROM customers

WHERE email LIKE '%@gmail.com';



j) Zobraz 3 nejprodávanější produkty podle celkového prodaného množství

SELECT products.product_name, SUM(order_details.quantity)

FROM products

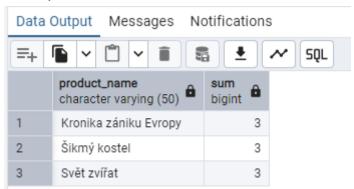
INNER JOIN order_details

ON products.product_id=order_details.product_id

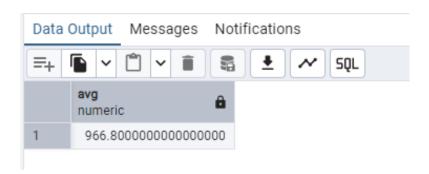
GROUP BY products.product_id

ORDER BY SUM(order_details.quantity) DESC

LIMIT 3;

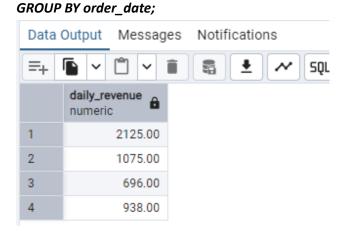


k) Zobraz průměrnou hodnotu objednávekSELECT AVG(total_price) from ORDERS;



I) Vypiš denní příjmy z objednávek

SELECT SUM(total_price) AS daily_revenue from ORDERS



m) Podíl tržeb podle kategorií produktů

SELECT products.category,

SUM(order_details.quantity*order_details.price_per_unit) AS category_revenue, ROUND ((SUM(order_details.quantity*order_details.price_per_unit)*100.0)/(SELECT SUM(quantity * price_per_unit) FROM order_details),2) AS revenue_percentage FROM products

INNER JOIN order_details

 $ON\ products.product_id = order_details.product_id$

GROUP BY products.category

ORDER BY revenue_percentage DESC;

