

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

КУРСОВА РОБОТА

з дисципліни “Бази даних”

спеціальність 121 – Програмна інженерія

на тему: Система аналізу цін споживчих товарів

Студентка
групи КП-93

Коротюк Марія
Ігорівна

(підпис)

Викладач
к.т.н, доцент кафедри
СПіСКС

Петрашенко А.В.

(підпис)

Захищено з оцінкою _____

Київ – 2020

АНОТАЦІЯ

Виконуючи курсову роботу ми створили програму для зберігання та аналізу поточних цін на товари у магазинах косметики та парфумерії за певною категорією. Дана система призначена для використання у сфері торгівлі та маркетингу.

Результатом роботи є можливість визначення цінових трендів, через порівняння середніх вартостей товарів однієї категорії у різних магазинах, також можна отримати перелік товарів, які є популярними серед клієнтів і у майбутньому користуватимуться ще більшим попитом, а також перелік товарів, які потребують додаткової реклами, адже мають гарні відгуки, але малу кількість продажів.

ЗМІСТ

Вступ.....	4
1. Аналіз інструментарію для виконання курсової роботи.....	5
2. Структура бази даних.....	6
3. Опис програмного забезпечення.....	7
3.1 Загальна структура програмного забезпечення.....	7
3.2 Опис модулів програмного забезпечення.....	7
3.3 Опис основних алгоритмів роботи.....	7
4. Аналіз функціонування засобів реплікації.....	13
5. Аналіз функціонування засобів резервування/відновлення.....	13
6. Аналіз результатів підвищення швидкодії запитів.....	13
7. Опис результатів аналізу предметної галузі.....	16
Висновки.....	17
Література.....	18
Додаток 1.....	19
Додаток 2.....	20
Додаток 3.....	21
Додаток 4.....	22
Додаток 5.....	23
Додаток 6.....	24

ВСТУП

Метою розробки є здобуття навичок використання СУБД та створення програмного забезпечення, яке буде взаємодіяти з реляційною базою даних, яка містить велику кількість даних, а також подальший аналіз цих даних і оформлення результатів у вигляді діаграм, графіків та таблиць.

Зокрема база даних містить актуальну інформацію про товари у магазинах косметики та парфумерії, таких як Prostor, Watsons, Brocard та Kosmo, дані про покупки та відгуки клієнтів, що дозволяє проаналізувати популярність товарів, та різницю середніх цін за категоріями у цих магазинах.

Кожен товар у базі даних належить до певної категорії, зокрема косметика для обличчя, очей та брів, губ та парфумерія. Оскільки більшість товарів у магазинах схожа та належить до однакових виробників, то ми аналізуємо середні ціни в межах певної категорії і визначаємо різницю у відсотках.

Дана робота є актуальною, адже дозволяє визначити цінові тренди, порівнюючи ціни у різних магазинах, і тому може використовуватись у сферах торгівлі та маркетингу, які зараз активно розвиваються.

Для кращого представлення більшість результатів аналізу також додатково зображуються як графіки та таблиці excel.

1. Аналіз інструментарію для виконання курсової роботи

Для виконання курсової роботи було обрано СУБД PostgreSQL, тому що ця СУБД підтримує бази даних необмеженого розміру, має потужні і надійні механізми транзакцій і реплікації, може бути встановлена на будь-якій операційній системі [1].

Також система є розширювана, оскільки можна створювати власні типи даних та індекси, а також розширювати поведінку за допомогою мов програмування [2].

Курсова робота написана мовою програмування python. Для генерації частини даних було використано фреймворк Scrapy, для аналізу даних бібліотека Data Science – pandas, для побудови графіків – бібліотека Matplotlib. Для взаємодії із базою даних за допомогою мови програмування python ми використали модуль psycopg2 та бібліотеку SQLAlchemy.

Scrapy – фреймворк для обходу сайтів та вилучення з них структурованих даних, які можуть використовуватися для широкого кола корисних застосувань: видобуток та обробка інформації чи історичне архівування [3].

pandas – програмна бібліотека, написана для мови програмування Python для маніпулювання даними та їхнього аналізу. Вона, зокрема, пропонує структури даних та операції для маніпулювання чисельними таблицями та часовими рядами [4].

Matplotlib – бібліотека на мові програмування Python для візуалізації даних двовимірною 2D графікою (3D графіка також підтримується). Отримувані зображення можуть бути використані як ілюстрації в публікаціях [5].

psycopg2 – найпопулярніший адаптер бази даних PostgreSQL, який є легким та ефективним у використанні [6].

SQLAlchemy - інструментарій SQL та об'єктно-реляційне відображення для мови програмування Python випущене під ліцензією MIT [7].

2. Структура бази даних

База даних містить наступні сутності (див. додаток 1):

1. store – сутність, яка містить дані про магазини, які будуть аналізуватись у роботі.
2. category – сутність, яка містить дані про категорії магазинів.
3. product – сутність, яка містить дані про товари, що належать до певного магазину та певної категорії.
4. purchase – сутність, яка містить дані про дату та час здійснення кожної покупки.
5. customer – сутність, яка містить дані користувача, який може створювати відгуки на товари.
6. review – сутність, яка містить дані про відгуки, створені певними користувачами на певні товари.

Між усіма таблицями зв'язки – це один до багатьох.

3. Опис програмного забезпечення

3.1 Загальна структура програмного забезпечення

Розроблене програмне забезпечення складається з двох частин: перша – програма для отримування даних з сайтів магазинів, друга – програма з консольним інтерфейсом для взаємодії з базою даних, проведення аналізу даних, здійснення пошуку, а також додатково було створено CRUD операції, що дозволяють додавати, видаляти і оновлювати інформацію у більшості таблиць.

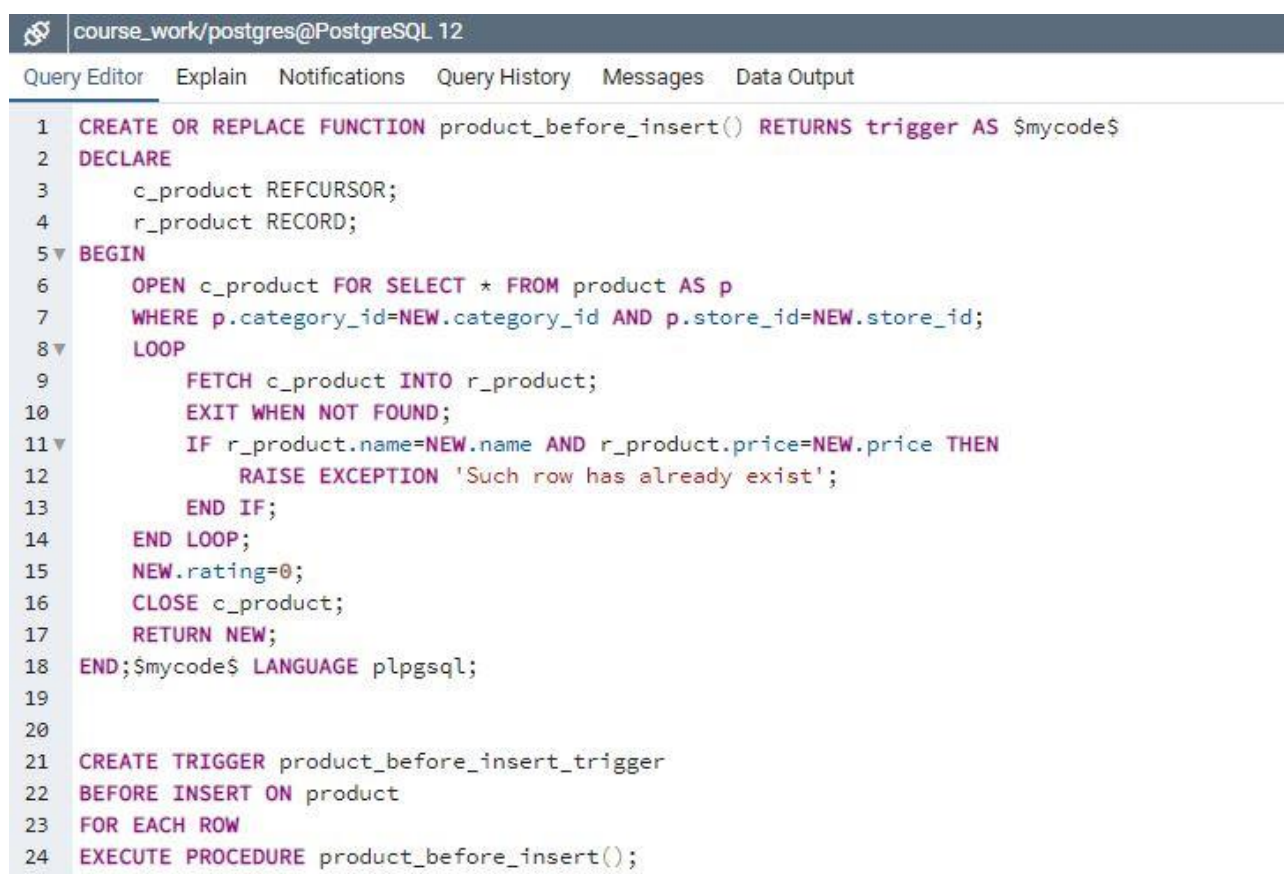
3.2 Опис модулів програмного забезпечення

Програмне забезпечення розроблене відповідно до структури MVC. У проекті є консольне меню, яке дозволяє обрати користувачу необхідну операцію. Програмне забезпечення містить один контролер, одне подання, окремі модулі для кожного класу, який відповідає певній сутності бази даних, зокрема store, category, product, purchase, customer, review. Також програма містить моделі, які відповідають сутностям (store_model, category_model, product_model, purchase_model, customer_model, review_model) та моделі для аналізу та пошуку даних (comparison_model і search_model). Моделі безпосередньо взаємодіють із базою даних і дозволяють отримувати програмі доступ до потрібної інформації.

3.3 Опис основних алгоритмів роботи.

Дані, що не генеруються за допомогою фреймворка Scrapy, генеруються спеціальними SQL запитами за допомогою моделей сутностей. Імена для таблиці customer створюються з використанням бібліотеки names.

CRUD операції виконуються з використанням бібліотеки SQLAlchemy, є валідація даних, які вводить користувач. Також перед додаванням даних у таблицю product здійснюється перевірка чи точно такого ж рядка не існує ще у таблиці. Для цього був написаний тригер (Рис. 1):



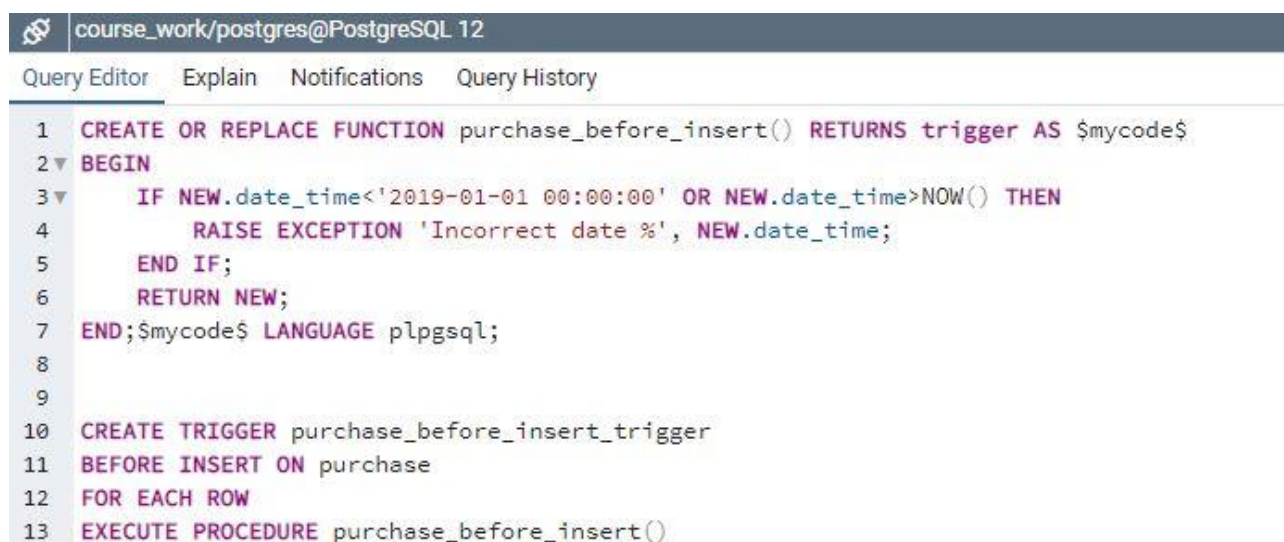
```

1 CREATE OR REPLACE FUNCTION product_before_insert() RETURNS trigger AS $mycode$
2 DECLARE
3     c_product REFCURSOR;
4     r_product RECORD;
5 BEGIN
6     OPEN c_product FOR SELECT * FROM product AS p
7     WHERE p.category_id=NEW.category_id AND p.store_id=NEW.store_id;
8 LOOP
9     FETCH c_product INTO r_product;
10    EXIT WHEN NOT FOUND;
11    IF r_product.name=NEW.name AND r_product.price=NEW.price THEN
12        RAISE EXCEPTION 'Such row has already exist';
13    END IF;
14 END LOOP;
15 NEW.rating=0;
16 CLOSE c_product;
17 RETURN NEW;
18 END;$mycode$ LANGUAGE plpgsql;
19
20
21 CREATE TRIGGER product_before_insert_trigger
22 BEFORE INSERT ON product
23 FOR EACH ROW
24 EXECUTE PROCEDURE product_before_insert();

```

Рис. 1 product_before_insert_trigger

Також були написані тригери, які перевіряють чи дати покупок, які мають бути додані у таблицю purchase при додаванні чи редагуванні, є не ранішими, ніж 01.01.2019 та не пізнішими, ніж поточна дата (Рис. 2):

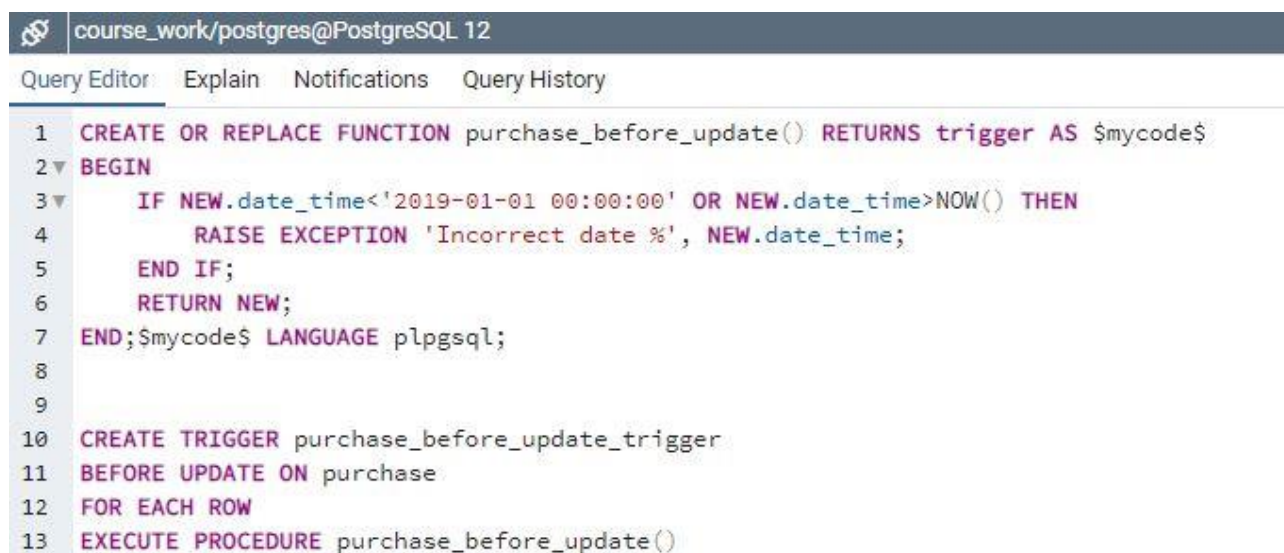


```

1 CREATE OR REPLACE FUNCTION purchase_before_insert() RETURNS trigger AS $mycode$
2 BEGIN
3     IF NEW.date_time<'2019-01-01 00:00:00' OR NEW.date_time>NOW() THEN
4         RAISE EXCEPTION 'Incorrect date %', NEW.date_time;
5     END IF;
6     RETURN NEW;
7 END;$mycode$ LANGUAGE plpgsql;
8
9
10 CREATE TRIGGER purchase_before_insert_trigger
11 BEFORE INSERT ON purchase
12 FOR EACH ROW
13 EXECUTE PROCEDURE purchase_before_insert();

```

Рис. 2 а) purchase_before_insert_trigger



```

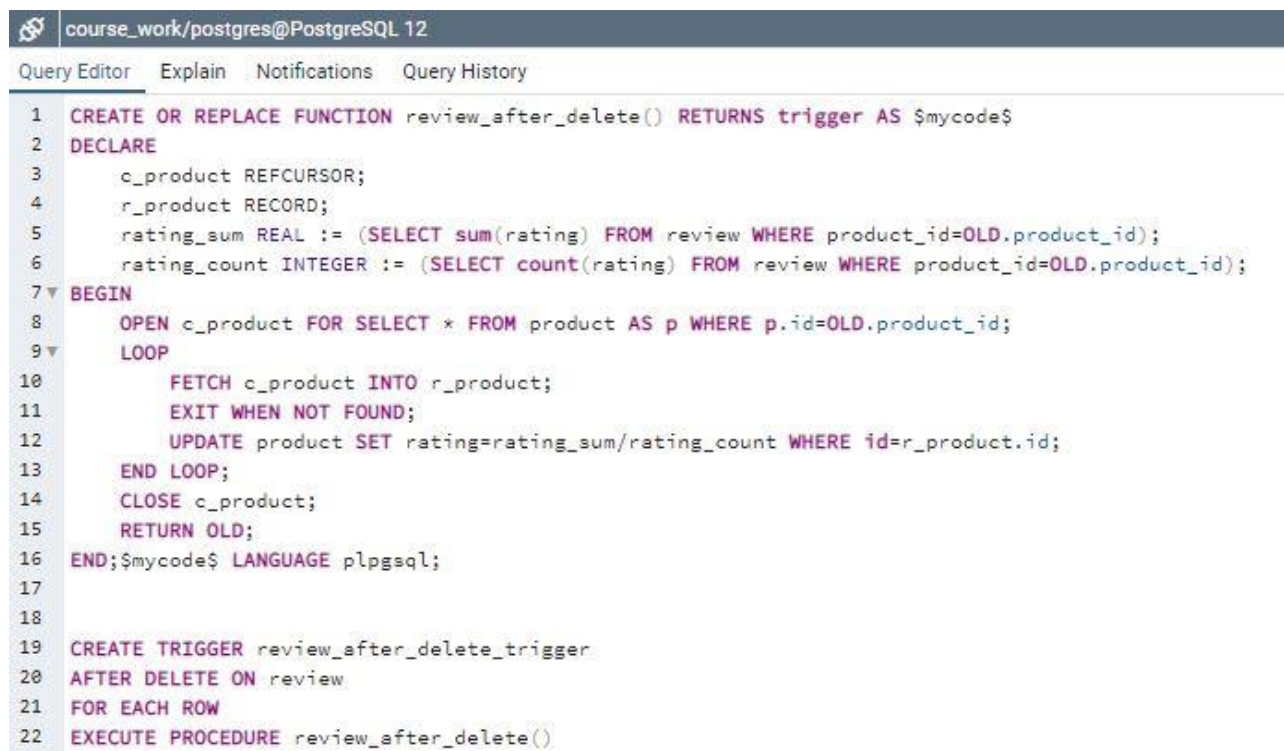
course_work/postgres@PostgreSQL 12
Query Editor Explain Notifications Query History

1 CREATE OR REPLACE FUNCTION purchase_before_update() RETURNS trigger AS $mycode$
2 BEGIN
3     IF NEW.date_time < '2019-01-01 00:00:00' OR NEW.date_time > NOW() THEN
4         RAISE EXCEPTION 'Incorrect date %', NEW.date_time;
5     END IF;
6     RETURN NEW;
7 END;$mycode$ LANGUAGE plpgsql;
8
9
10 CREATE TRIGGER purchase_before_update_trigger
11 BEFORE UPDATE ON purchase
12 FOR EACH ROW
13 EXECUTE PROCEDURE purchase_before_update()

```

Рис. 2 б) purchase_after_insert_trigger

Було створено тригер, який після додавання, редагування чи видалення запису у таблиці review обраховує і перезаписує середнє значення у полі rating таблиці product та перед додаванням, редагуванням у таблиці review здійснюється перевірка чи значення rating знаходиться в межах від 1 до 5 (Рис. 3):



```

course_work/postgres@PostgreSQL 12
Query Editor Explain Notifications Query History

1 CREATE OR REPLACE FUNCTION review_after_delete() RETURNS trigger AS $mycode$
2 DECLARE
3     c_product REFCURSOR;
4     r_product RECORD;
5     rating_sum REAL := (SELECT sum(rating) FROM review WHERE product_id=OLD.product_id);
6     rating_count INTEGER := (SELECT count(rating) FROM review WHERE product_id=OLD.product_id);
7 BEGIN
8     OPEN c_product FOR SELECT * FROM product AS p WHERE p.id=OLD.product_id;
9     LOOP
10         FETCH c_product INTO r_product;
11         EXIT WHEN NOT FOUND;
12         UPDATE product SET rating=rating_sum/rating_count WHERE id=r_product.id;
13     END LOOP;
14     CLOSE c_product;
15     RETURN OLD;
16 END;$mycode$ LANGUAGE plpgsql;
17
18
19 CREATE TRIGGER review_after_delete_trigger
20 AFTER DELETE ON review
21 FOR EACH ROW
22 EXECUTE PROCEDURE review_after_delete()

```

Рис. 3 а) review_after_delete_trigger

```

course_work/postgres@PostgreSQL 12
Query Editor Query History Data Output Explain Notifications Messages
1 CREATE OR REPLACE FUNCTION review_after_insert() RETURNS trigger AS $mycode$
2 DECLARE
3     c_product REFCURSOR;
4     r_product RECORD;
5     rating_sum REAL := (SELECT sum(rating) FROM review WHERE product_id=NEW.product_id);
6     rating_count INTEGER := (SELECT count(rating) FROM review WHERE product_id=NEW.product_id);
7 BEGIN
8     OPEN c_product FOR SELECT * FROM product AS p WHERE p.id=NEW.product_id;
9     LOOP
10         FETCH c_product INTO r_product;
11         EXIT WHEN NOT FOUND;
12         UPDATE product SET rating=rating_sum/rating_count WHERE id=r_product.id;
13     END LOOP;
14     CLOSE c_product;
15     RETURN NEW;
16 END;$mycode$ LANGUAGE plpgsql;
17
18
19 CREATE TRIGGER review_after_insert_trigger
20 AFTER INSERT ON review
21 FOR EACH ROW
22 EXECUTE PROCEDURE review_after_insert();

```

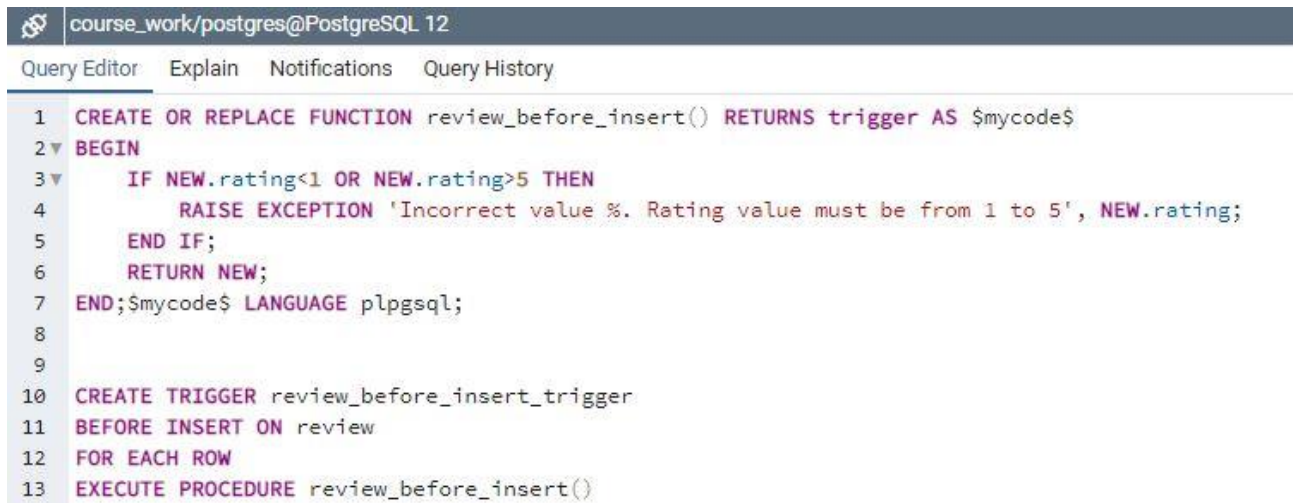
Рис. 3 б) review_after_insert_trigger

```

course_work/postgres@PostgreSQL 12
Query Editor Explain Notifications Query History
1 CREATE OR REPLACE FUNCTION review_after_update() RETURNS trigger AS $mycode$
2 DECLARE
3     old_c_product REFCURSOR;
4     old_r_product RECORD;
5     new_c_product REFCURSOR;
6     new_r_product RECORD;
7     old_rating_sum REAL := (SELECT sum(rating) FROM review WHERE product_id=OLD.product_id);
8     old_rating_count INTEGER := (SELECT count(rating) FROM review WHERE product_id=OLD.product_id);
9     new_rating_sum REAL := (SELECT sum(rating) FROM review WHERE product_id=NEW.product_id);
10    new_rating_count INTEGER := (SELECT count(rating) FROM review WHERE product_id=NEW.product_id);
11 BEGIN
12     OPEN old_c_product FOR SELECT * FROM product AS p WHERE p.id=OLD.product_id;
13     LOOP
14         FETCH old_c_product INTO old_r_product;
15         EXIT WHEN NOT FOUND;
16         UPDATE product SET rating=old_rating_sum/old_rating_count WHERE id=old_r_product.id;
17     END LOOP;
18     CLOSE old_c_product;
19
20     OPEN new_c_product FOR SELECT * FROM product AS p WHERE p.id=NEW.product_id;
21     LOOP
22         FETCH new_c_product INTO new_r_product;
23         EXIT WHEN NOT FOUND;
24         UPDATE product SET rating=new_rating_sum/new_rating_count WHERE id=new_r_product.id;
25     END LOOP;
26     CLOSE new_c_product;
27     RETURN NEW;
28 END;$mycode$ LANGUAGE plpgsql;
29
30 CREATE TRIGGER review_after_update_trigger
31 AFTER UPDATE ON review
32 FOR EACH ROW
33 EXECUTE PROCEDURE review_after_update();

```

Рис. 3 в) review_after_update_trigger

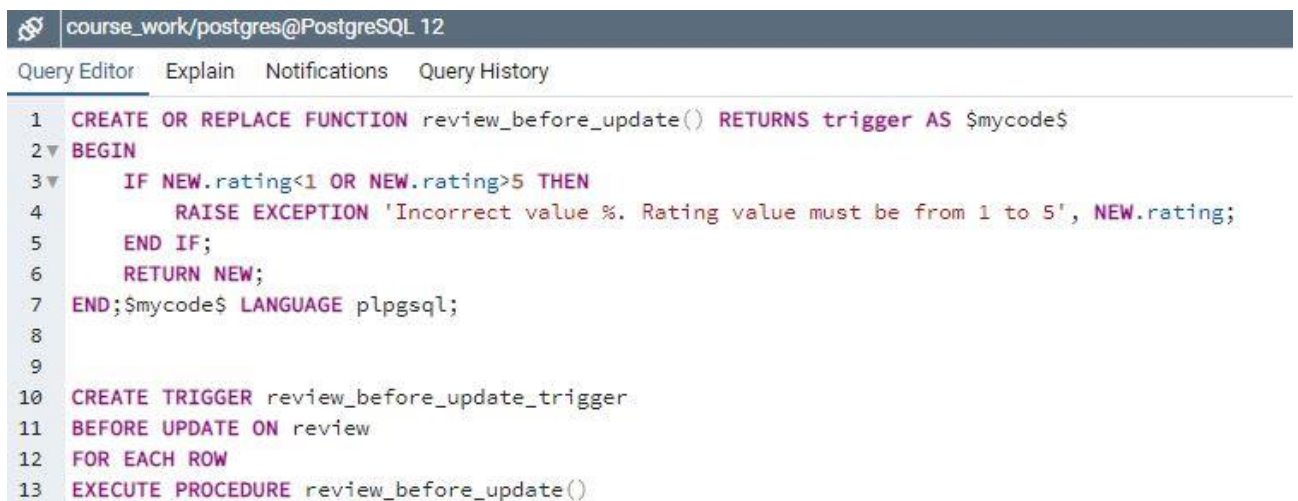


```

1 CREATE OR REPLACE FUNCTION review_before_insert() RETURNS trigger AS $mycode$
2 BEGIN
3     IF NEW.rating<1 OR NEW.rating>5 THEN
4         RAISE EXCEPTION 'Incorrect value %. Rating value must be from 1 to 5', NEW.rating;
5     END IF;
6     RETURN NEW;
7 END;$mycode$ LANGUAGE plpgsql;
8
9
10 CREATE TRIGGER review_before_insert_trigger
11 BEFORE INSERT ON review
12 FOR EACH ROW
13 EXECUTE PROCEDURE review_before_insert()

```

Рис. 3 г) review_before_insert_trigger



```

1 CREATE OR REPLACE FUNCTION review_before_update() RETURNS trigger AS $mycode$
2 BEGIN
3     IF NEW.rating<1 OR NEW.rating>5 THEN
4         RAISE EXCEPTION 'Incorrect value %. Rating value must be from 1 to 5', NEW.rating;
5     END IF;
6     RETURN NEW;
7 END;$mycode$ LANGUAGE plpgsql;
8
9
10 CREATE TRIGGER review_before_update_trigger
11 BEFORE UPDATE ON review
12 FOR EACH ROW
13 EXECUTE PROCEDURE review_before_update()

```

Рис. 3 д) review_before_update_trigger

Порівняння проданих продуктів відбувається наступним чином: визначається кількість проданих товарів у кожній категорії для кожного магазину і ці результати записуються у таблицю за допомогою бібліотеки pandas. Далі створюються графіки чи гістограми та записуються отримані дані у Excel файл. Для визначення різниці цін спочатку визначаються 2 магазини, для яких буде проводитись аналіз, далі знаходяться середні ціни за категорією у відповідному магазині та записуються у таблицю, створюються графіки та здійснюється запис у Excel файл. Потім вираховується різниця у відсотках і дана інформація виводиться для користувача у консолі.

Для прогнозування популярності товарів потрібно визначити які товари купують частіше, які мають кількість відгуків більше середнього та їх рейтинг є

не нижчим 4. Саме це і товари є найпопулярнішими і клієнти задоволені їх якістю, отже попит на них буде продовжувати рости.

Для визначення переліку товарів, які потребують додаткової реклами, потрібно знайти товари які купують рідко і вони мають мало відгуків проте їх рейтинг є не нижчим 4.2, що свідчить про те, що клієнти, які їх купували ними задоволені. Отже, такі товари потрібно рекламувати, щоб збільшити аудиторію їх споживачів і зробити їх популярнішими.

У програмі також є можливість пошуку записів у таблиці product за рейтингом у визначеному користувачем діапазоні, та пошук у таблиці purchase за датою – також у визначеному користувачем діапазоні,

4. Аналіз функціонування засобів реплікації

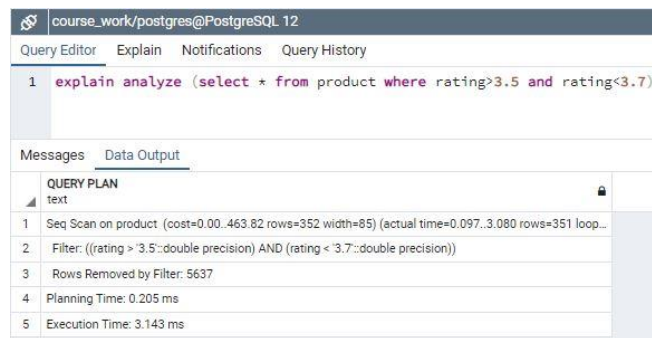
При виконанні курсової роботи реплікація бази даних не була реалізована.

5. Аналіз функціонування засобів резервування/відновлення

При виконанні курсової роботи резервування/відновлення бази даних не були реалізовані.

6. Аналіз результатів підвищення швидкодії запитів

Для запитів, які використовуються для пошуку, були використані індекси для підвищення їх швидкодії. В обох випадках були застосовані індекси BTree, адже пошук здійснюється у заданому діапазоні з використанням знаків < > в умові (Рис. 4):



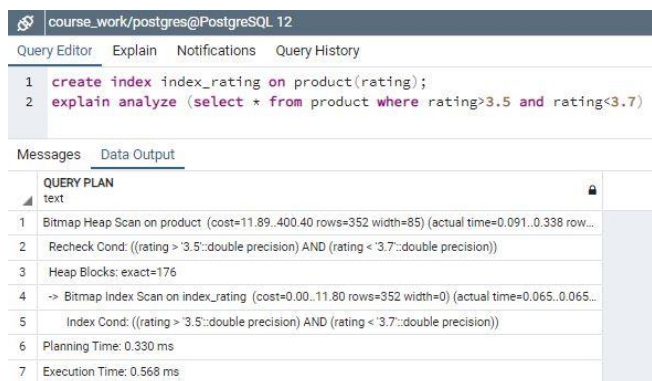
The screenshot shows the PostgreSQL Query Editor interface. The query editor contains the following SQL query:

```
1 explain analyze (select * from product where rating>3.5 and rating<3.7)
```

The 'Data Output' tab is selected, displaying the 'QUERY PLAN' for the query. The plan consists of the following steps:

Step	Operation	Cost	Rows	Width	Actual Time	Rows
1	Seq Scan on product	(cost=0.00..463.82)	rows=352	width=85	(actual time=0.097..3.080)	rows=351
2	Filter: ((rating > 3.5::double precision) AND (rating < 3.7::double precision))					
3	Rows Removed by Filter:		5637			
4	Planning Time:	0.205 ms				
5	Execution Time:	3.143 ms				

Рис. 4 а) Приклад запиту до створення індекса для таблиці product



The screenshot shows the PostgreSQL Query Editor interface. The query editor contains the following SQL queries:

```
1 create index index_rating on product(rating);
2 explain analyze (select * from product where rating>3.5 and rating<3.7)
```

The 'Data Output' tab is selected, displaying the 'QUERY PLAN' for the query. The plan consists of the following steps:

Step	Operation	Cost	Rows	Width	Actual Time	Rows
1	Bitmap Heap Scan on product	(cost=11.89..400.40)	rows=352	width=85	(actual time=0.091..0.338)	row...
2	Recheck Cond: ((rating > 3.5::double precision) AND (rating < 3.7::double precision))					
3	Heap Blocks: exact=176					
4	-> Bitmap Index Scan on index_rating	(cost=0.00..11.80)	rows=352	width=0	(actual time=0.065..0.065...)	
5	Index Cond: ((rating > 3.5::double precision) AND (rating < 3.7::double precision))					
6	Planning Time:	0.330 ms				
7	Execution Time:	0.568 ms				

Рис. 4 б) Приклад запиту після створення індекса для таблиці product

course_work/postgres@PostgreSQL 12

Query Editor

Explain

Notifications

Query History

1

explain analyze (select * from purchase where

2

date_time>'2020-12-12' and date_time<'2020-12-14')

Messages

Data Output

QUERY PLAN

text

1 Seq Scan on purchase (cost=0.00..2063.85 rows=261 width=16) (actual time=0.115...

2 Filter: ((date_time > '2020-12-12 00:00:00':timestamp without time zone) AND (date...

3 Rows Removed by Filter: 100868

4 Planning Time: 0.108 ms

5 Execution Time: 27.747 ms

Рис. 4 в) Приклад запиту до створення індекса для таблиці purchase

course_work/postgres@PostgreSQL 12	
Query Editor	Explain Notifications Query History
<pre>1 create index index_date on purchase(date_time); 2 explain analyze (select * from purchase where 3 date_time>'2020-12-12' and date_time<'2020-12-14')</pre>	
Messages	Data Output
	<div>QUERY PLAN</div> <div>text</div> <div>1 Bitmap Heap Scan on purchase (cost=6.96..461.71 rows=260 width=16) (actual tim...</div> <div>2 Recheck Cond: ((date_time > '2020-12-12 00:00:00':timestamp without time zone) A...</div> <div>3 Heap Blocks: exact=203</div> <div>4 -> Bitmap Index Scan on index_date (cost=0.00..6.89 rows=260 width=0) (actual ti...</div> <div>5 Index Cond: ((date_time > '2020-12-12 00:00:00':timestamp without time zone) ...</div> <div>6 Planning Time: 0.152 ms</div> <div>7 Execution Time: 0.379 ms</div>

Рис. 4 г) Приклад запиту після створення індекса для таблиці purchase

Зокрема швидкість пошуку у таблиці product збільшилась у 3 рази з використанням індексів, для цього ми порівняли швидкість виконання до створення і після створення 10 разів (Рис 5 (а)), а у таблиці purchase відбулось пришвидшення приблизно у 50 разів, порівняння також проводилось 10 разів (Рис 5 (б)). Гістограми порівняння швидкодії запитів наведені у Додатку 2.

	A	B
1	Before index_rating	After index_rating
2	2,49	0,72
3	2,35	0,74
4	3,52	0,39
5	3,31	0,49
6	3,63	0,80
7	2,38	0,67
8	3,09	0,62
9	2,67	0,44
10	3,10	0,63
11	2,91	0,43

Рис. 5 а) Порівняння швидкодії запиту для пошуку у таблиці product

	A	B
1	Before index_date	After index_date
2	20,11	0,34
3	22,09	0,32
4	19,17	0,55
5	28,98	0,31
6	31,38	0,57
7	21,74	0,49
8	26,96	0,56
9	22,98	0,33
10	22,29	0,3
11	27,74	0,35

Рис. 5 б) Порівняння швидкодії запиту для пошуку у таблиці purchase

7. Опис результатів аналізу предметної галузі

При порівнянні кількості проданих товарів за категоріями у різних магазинах за 12.12.2020 отримуємо, що у трьох з чотирьох категорій найбільша кількість проданих товарів належить магазину Brocard. Найбільша кількість проданих товарів у цей день – 18, а найменша – 2. Також можемо порівняти кількості продажів за місяць і отримати їх графік (Додаток 3).

При порівнянні середніх цін на товари у магазинах Brocard і Watsons отримуємо, що в Brocard ціни в рази вищі, особливо різняться ціни на парфумерію (Додаток 4).

У Додатках 5 та 6 наведені гістограми, які характеризують рейтинги популярних та не популярних товарів та їх кількість продажів та відгуків за категоріями.

Висновки

Отже, у курсовій роботі ми створили програмне забезпечення, яке дозволяє отримувати актуальні дані з інтернет-магазинів Prostor, Watsons, Brocard та Kosmo, записувати ці дані у реляційну базу даних PostgreSQL, проводити аналіз цих даних за категоріями із створенням таблиць, графіків та гістограм, та отримувати перелік популярних товарів і непопулярних, які потребують додаткової реклами. Тому дане програмне забезпечення може використовуватись у сфері торгівлі та маркетингу.

Для написання програми ми використали фреймворк Scrapy, модуль psycopg2, бібліотеки pandas, matplotlib та SQLAlchemy.

Також програма дозволяє генерувати дані за допомогою SQL запитів і здійснювати пошук, який виконується швидше завдяки використанню індексів.

Література

1. What is PostgreSQL? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.educba.com/what-is-postgresql/>.
2. Chapter 11. Indexes [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/9.1/indexes-intro.html>.
3. Створення веб павука з Python та Scrapy [Електронний ресурс] – Режим доступу до ресурсу: <https://codeguida.com/post/254/>.
4. pandas [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Pandas>.
5. matplotlib [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Matplotlib>.
6. Доступ к базе данных PostgreSQL с помощью psycopg2 [Електронний ресурс] – Режим доступу до ресурсу: <https://riptutorial.com/ru/python/example/18257/доступ-к-базе-данных-postgresql-с-помощью-psycopg2>.
7. SQLAlchemy [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/SQLAlchemy>.

ДОДАТОК 1

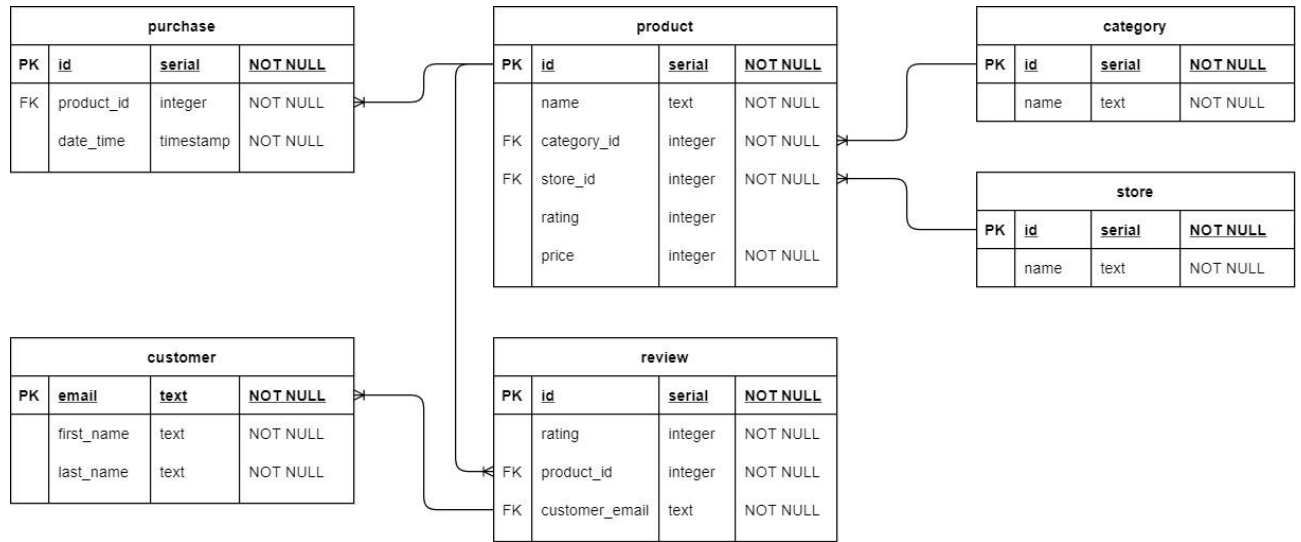


Рис. Структура бази даних

ДОДАТОК 2

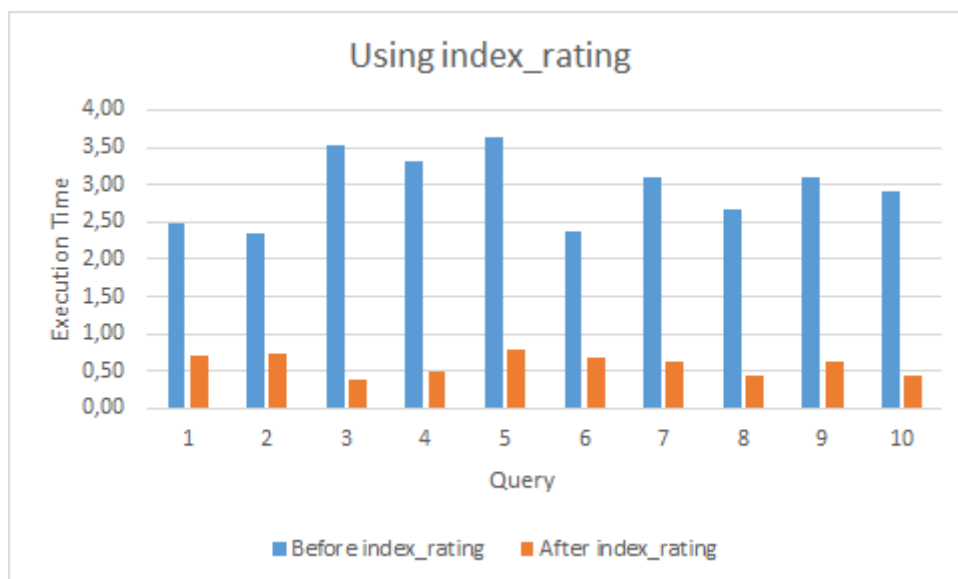


Рис. Порівняння швидкодії запитів для таблиці product

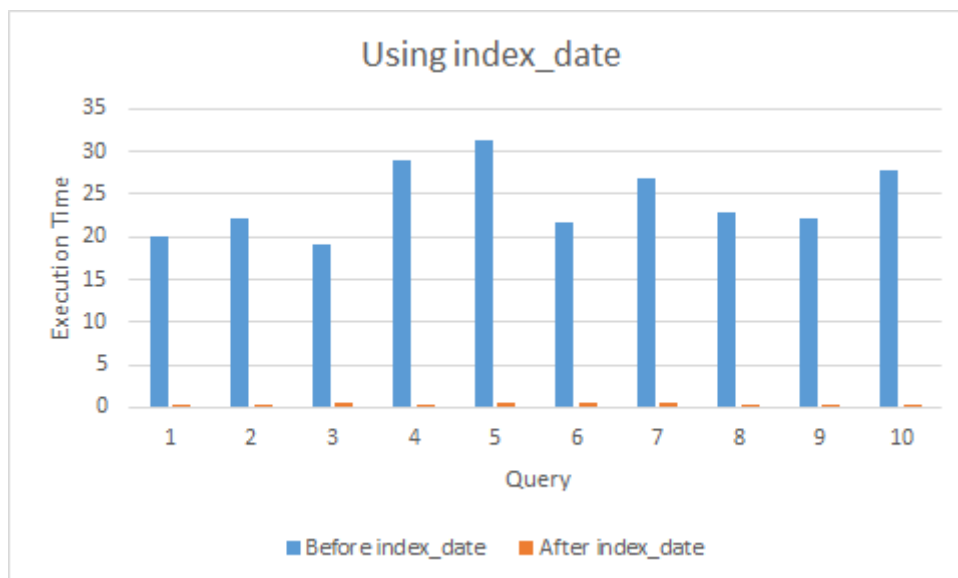


Рис. Порівняння швидкодії запитів для таблиці purchase

ДОДАТОК 3

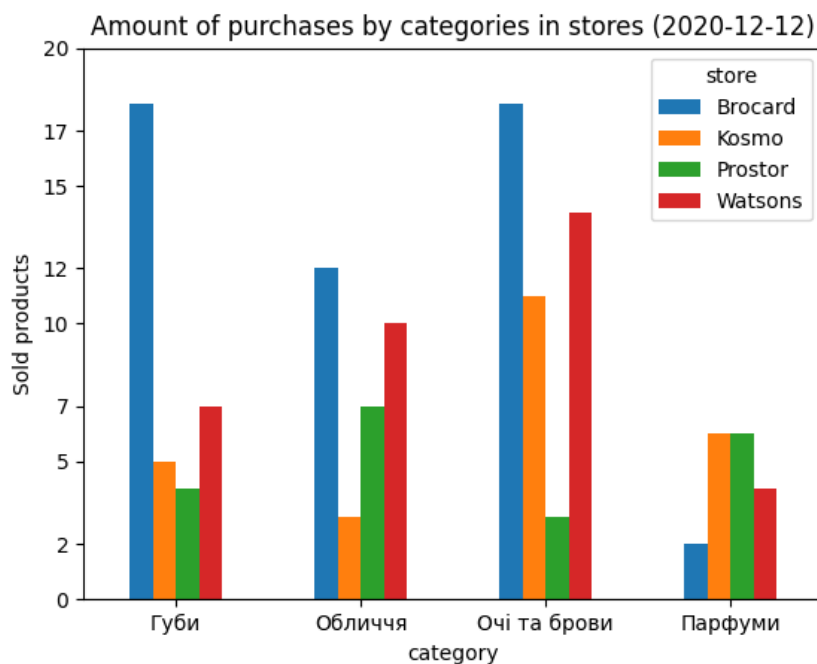


Рис. Приклад порівняння кількості покупок за категоріями у різних магазинах за певний день

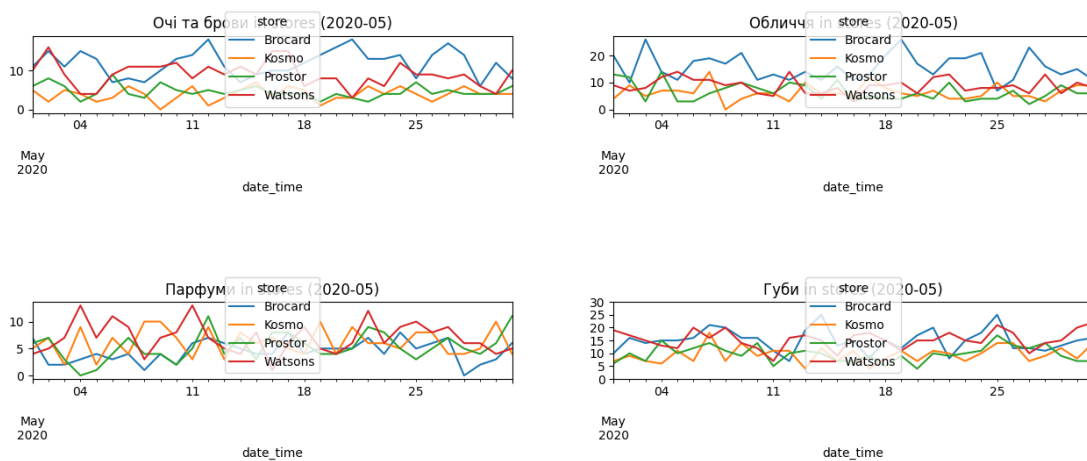


Рис. Приклад порівняння кількості покупок за категоріями у різних магазинах за певний місяць

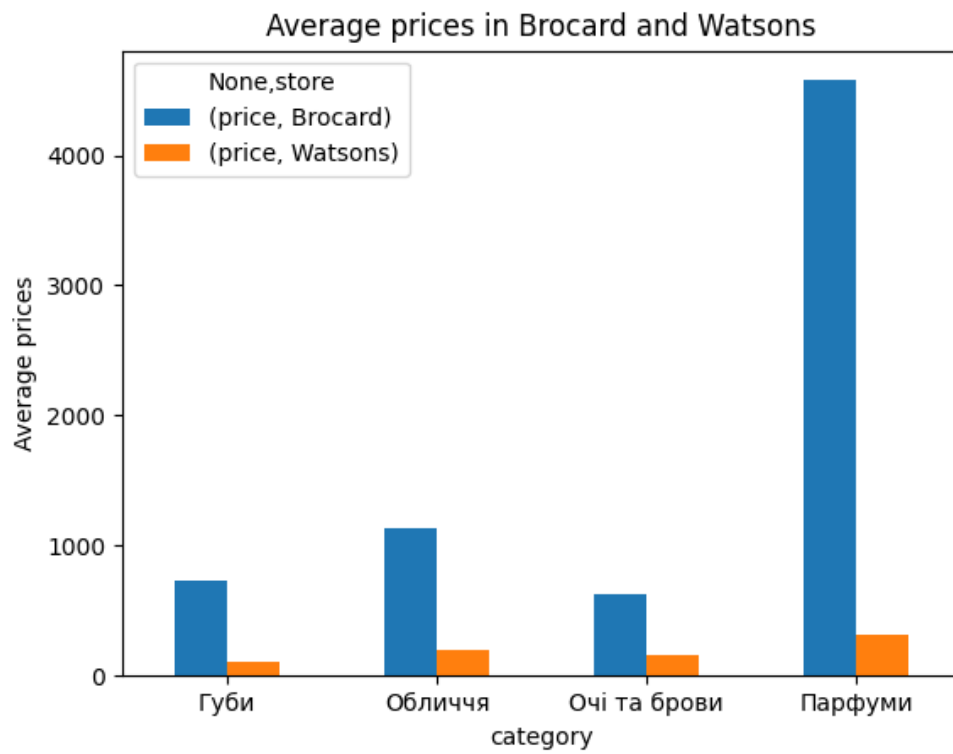


Рис. Приклад порівняння середніх цін за категоріями у двох магазинах

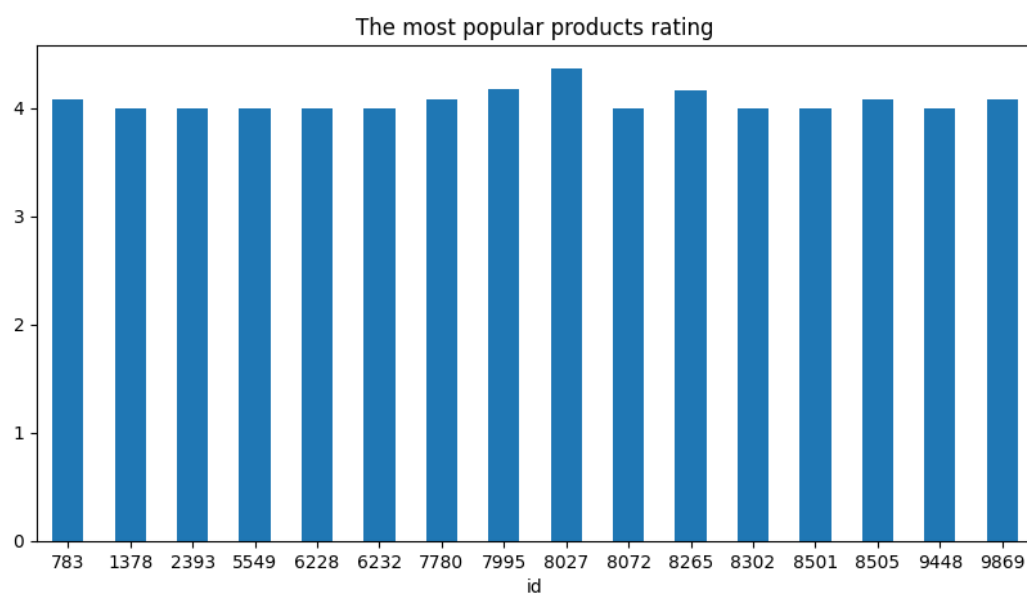


Рис. Приклад порівняння популярних товарів за рейтингом

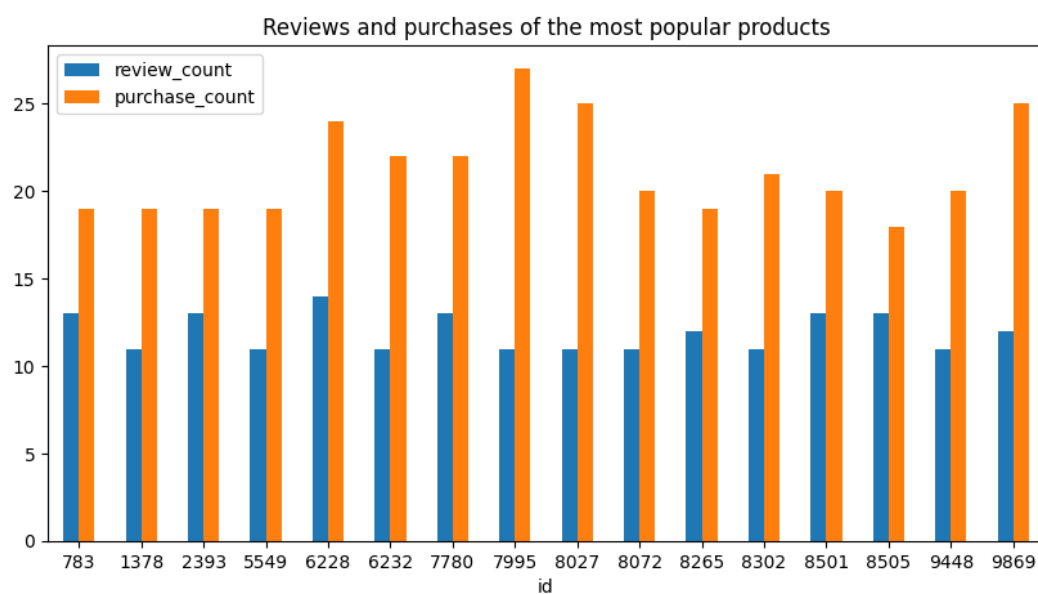


Рис. Приклад порівняння популярних товарів за кількістю покупок та відгуків

ДОДАТОК 6

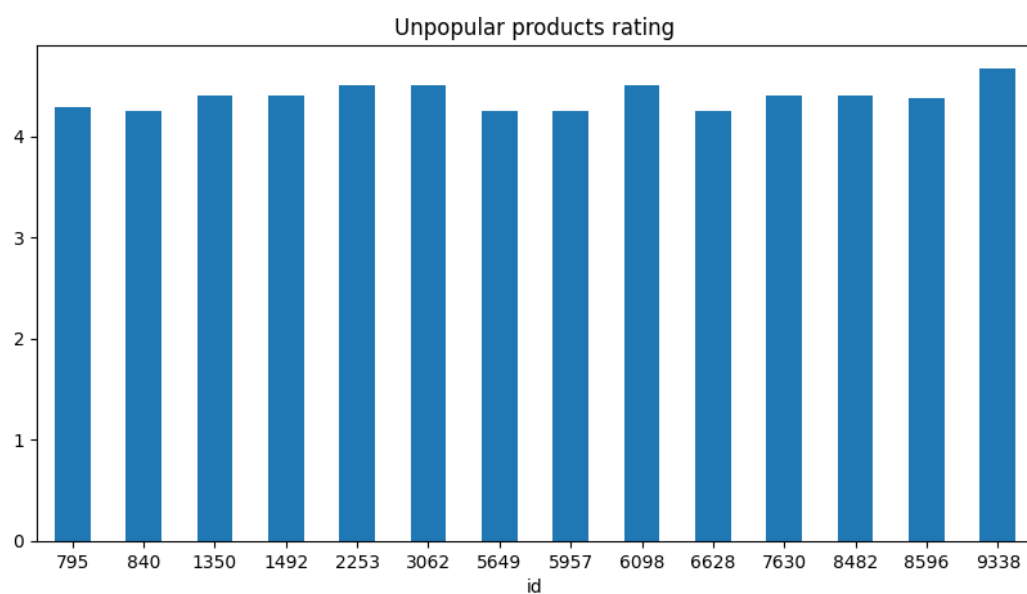


Рис. Приклад порівняння непопулярних товарів за рейтингом

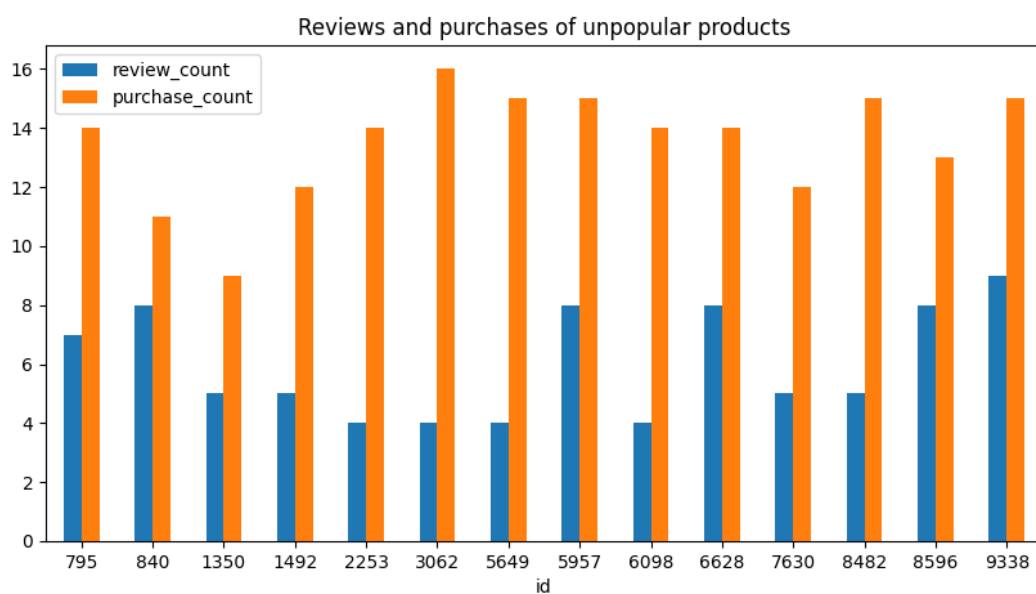


Рис. Приклад порівняння непопулярних товарів за кількістю покупок та відгуків