

BeautyHive

PROJECT DELIVERABLE #3

By Julia Cardillo // CS673



The Issue

- Finding credible reviews for makeup and skincare products can be difficult.
- Reviews are either misleading or are for fake products sold by 3rd party entities.



Sponsored · Shop beauty of joseon

Product	Price	Store	Rating	Reviews	Special offer
[Top Pick] BEAUTY OF JOSEON - Relief Sun : Rice + ...	\$14.40	Stylevana	★★★★★	(744)	Special offer
Beauty of Joseon Relief Sun 50ml	\$14.40	YesStyle.com	★★★★★	(85,934)	Special offer
Beauty of Joseon Relief Sun: Rice + Probiotics Double...	\$7.16	Walmart	★★★★★	(85,934)	Free shipping
Beauty of Joseon Relief Sun: Rice + Probiotics, SPF 50...	\$14.40	iHerb	★★★★★	(85,934)	
BEAUTY OF JOSEON - Relief Sun : Rice +...	\$57.60	Stylevana	★★★★★	(744)	Special offer
Beauty of Joseon Relief Sun: Rice + Probiotics (SPF50...	\$33.00	Olive Young Global			

Double Sun Protection (Sunset & water resistant)

Beauty of Joseon
Beauty of Joseon Relief Sun: Rice + Probiotics Double Set, Sunscreen
★★★★★ (85,934)

Free 30-day returns

Now \$7.16 (was \$14.40)
You save \$4.82

Price when purchased online

Add to cart

How do you want your items?

Shipping arrives Jul 2 Free
Pickup Not available
Delivery Not available

Delivery to Sacramento, 95829

Sold and shipped by BLOCCALL
View seller information | Visit seller store

Free 30-day returns Details

Add to list | Add to registry

More seller options (5)
Starting from \$7.00
Compare all sellers

Similar items you might like
Based on what customers bought

Product	Price	Store	Rating	Reviews	Special offer
Beauty of Joseon Relief Sun : Rice + Probiotics Sunscreen SPF 50+ PA++++, 50ml (2-...	Now \$10.50 (was \$14.40) More options from \$7.00	Stylevana	★★★★★	(744)	
Beauty of Joseon Relief Sun: Rice + Probiotics Sunscreen SPF 50+ PA++++, 50ml (2-...	Now \$7.19 (was \$14.40) Shipping arrives in 3-5 days	YesStyle.com	★★★★★	(85,934)	
Beauty of Joseon - Rice Probiotics Relief Sun Mini SPF50 PA+ (30ml)	\$6.49 Shipping arrives in 3-5 days	Walmart	★★★★★	(85,934)	
Sunscreen, Korean skin care, Korean sunscreen, Sunscreen SPF 50+, Korean Skin Care...	Now \$26.7 (was \$33.00) Options from \$27 - \$33.00	Olive Young Global	★★★★★	(744)	

Double Sun Protection (Sunset & water resistant)
Now \$14.65 (was \$14.40)
Double Sun Protection Clear Tinted Face Sunscreen SPF 46 Textured Sunscreen 1.7oz
In stock shipping
Add

My Solution

- Introducing... BeautyHive!
- Hub for all beauty product reviews.
- Mobile application that centralizes and aggregates reviews, made for beauty enthusiasts by beauty enthusiasts.



WELCOME
Please sign in:

USERNAME:

PASSWORD



Deliverable #1 - Design and Data Requirements

DATA GENERATION

Data is generated when users register for the platform and create product reviews.

DATA TYPES

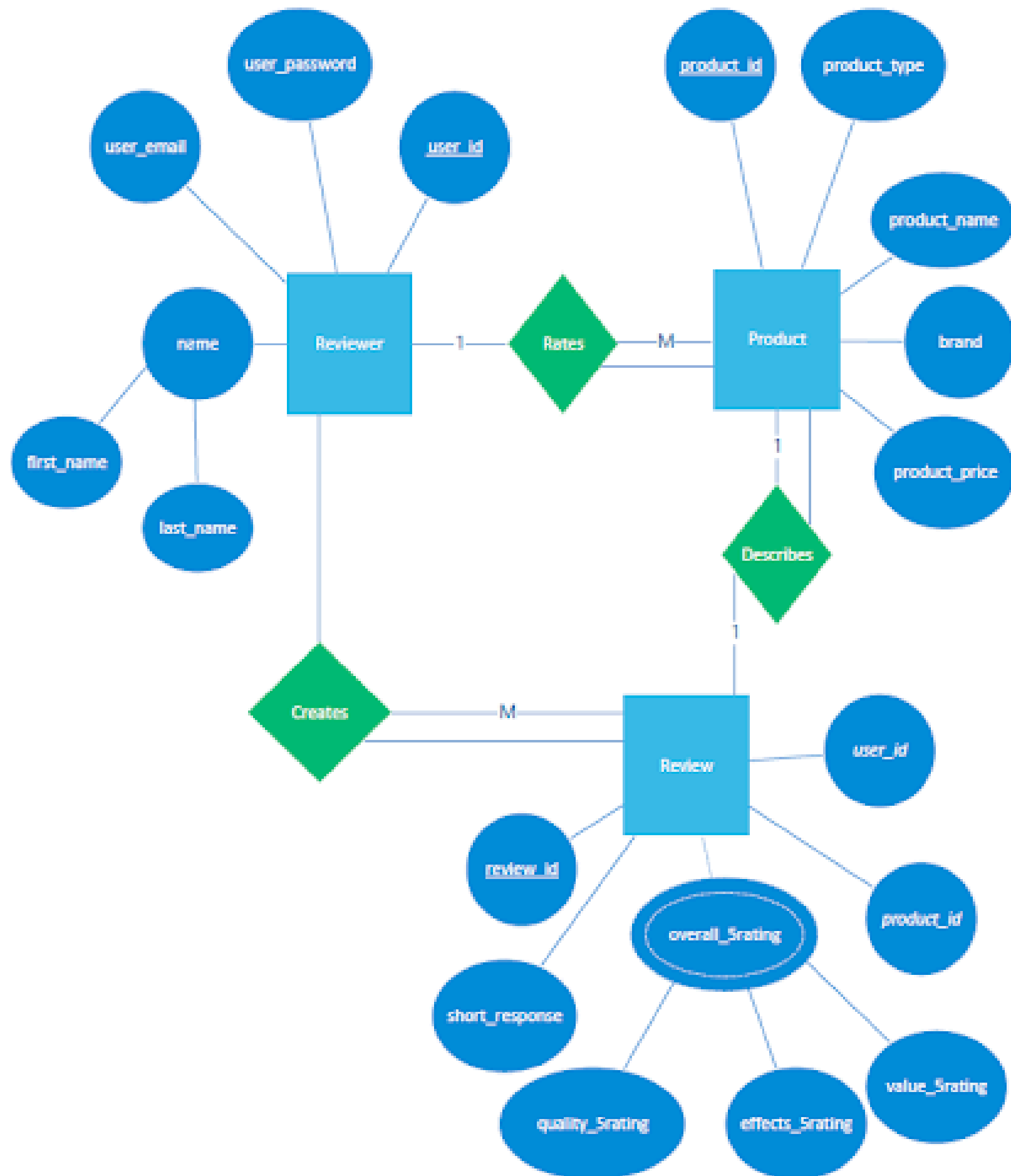
- Quantitative (product ratings)
- Qualitative (short responses)
- Categorical data (product type)

RELATIONAL DATABASE

- Data is structured
- Data is regularly queried
- Data can be easily manipulated
- Has constraints
- ACID properties



ER Diagram and Relational Schema



Relational Schema

Reviewer (user_id, user_password, user_email, first_name, last_name)

Review (review_id, user_id, product_id, overall_5rating, quality_5rating, effects_5rating, value_5rating)

Product (product_id, product_type, product_name, brand, product_price)



Deliverable #2 - Definitions of Terms Used

USER

An individual who registers for this platform, with the purpose of either viewing beauty product reviews or creating and posting their product reviews.

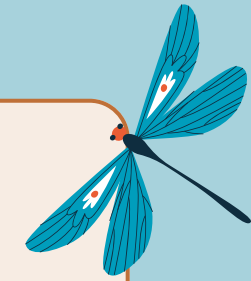
PRODUCT

A beauty (makeup or skincare) product that is available for sale online and in-person via various retailers. These products are reviewed on BeautyHive.

REVIEW

A User's comprehensive rating of a product, including numerical and short responses. A user can only create one review per product.





Changes Made to Deliverable #1

Product and Review cardinality made to be one-to-many, removed relationship from user to product

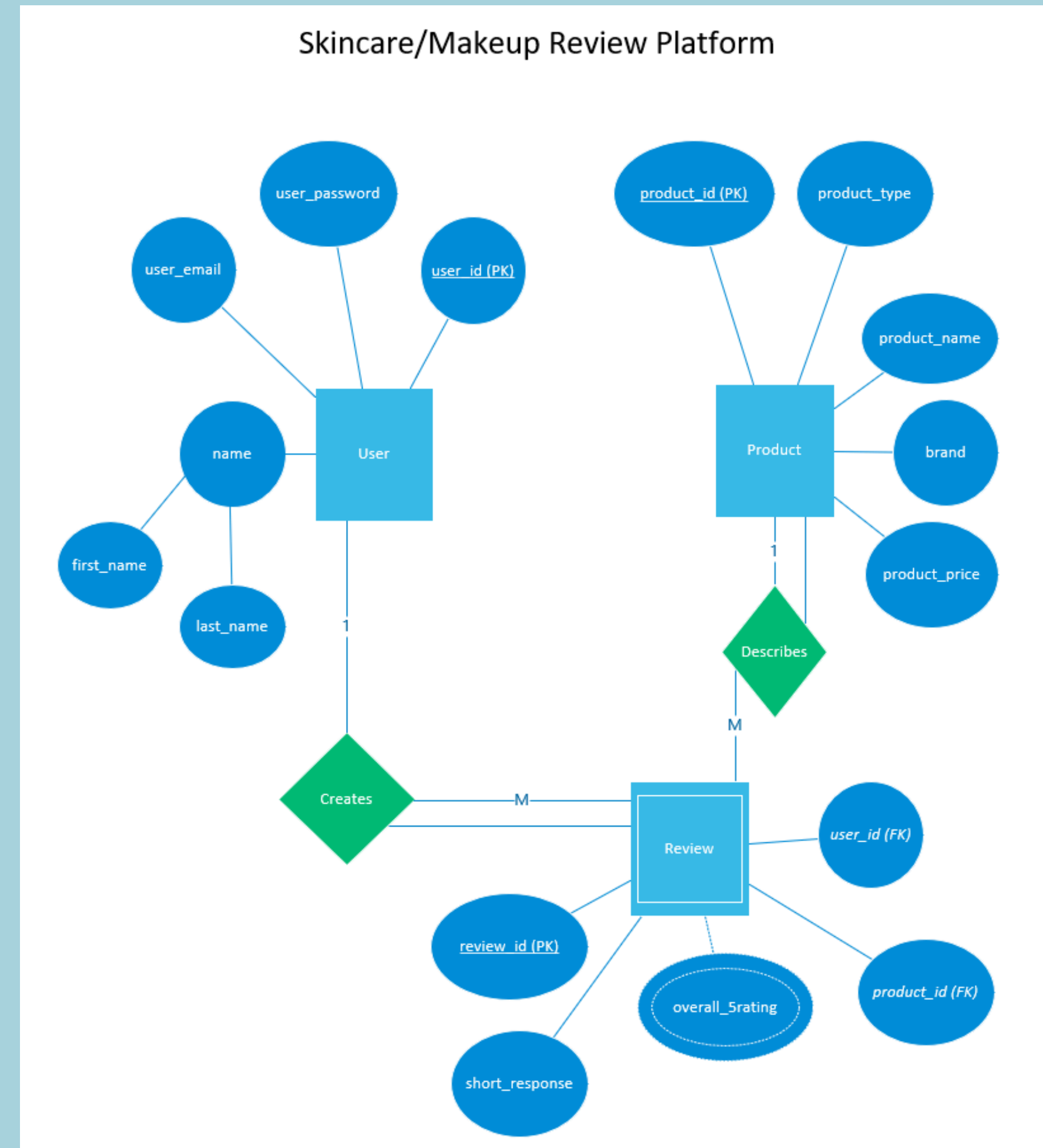
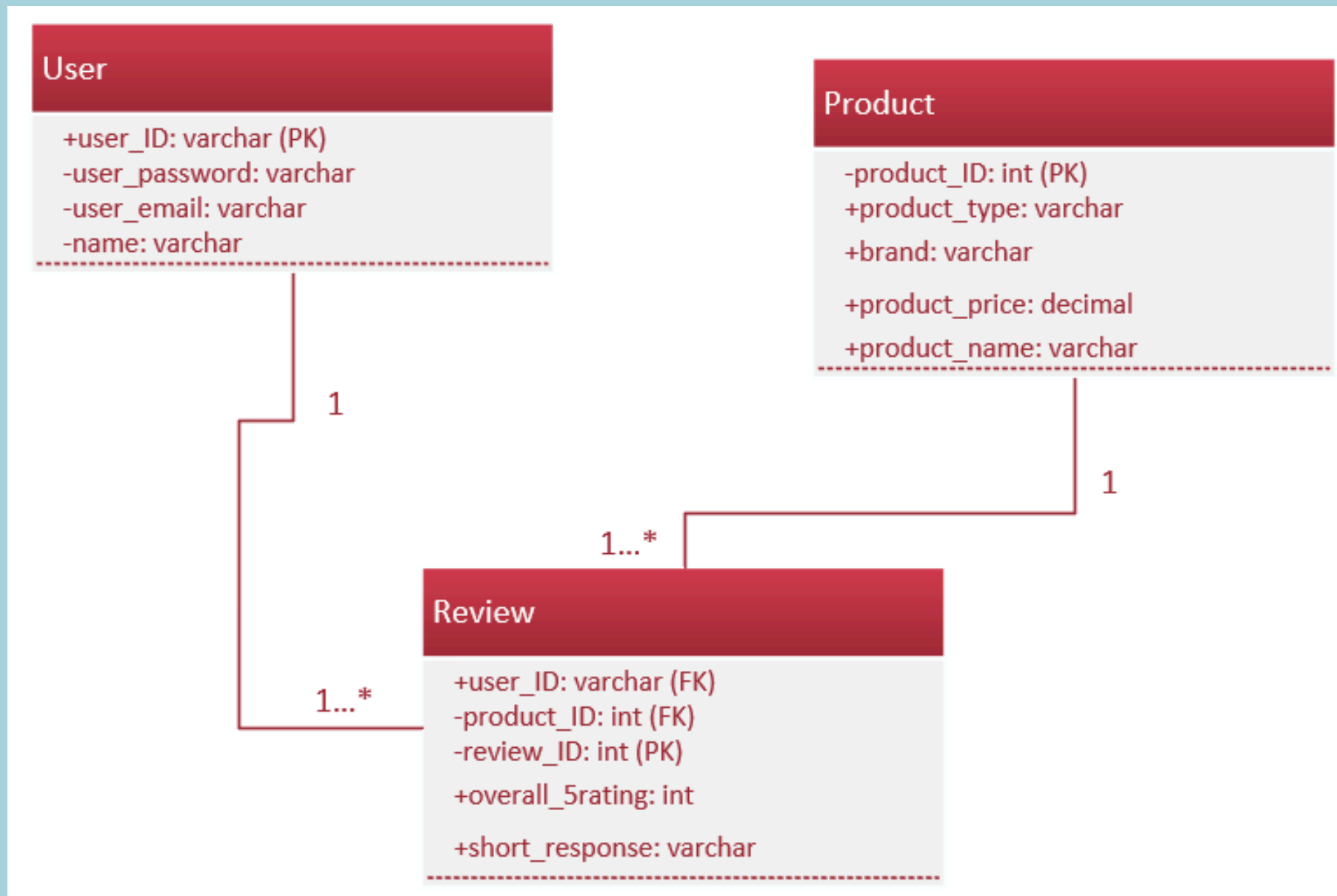
Product and User are strong entities, while Review is a weak entity

Changed Reviewer table to User for clarity

Clarified Primary and Foreign Keys



Updated ER Diagram and UML Diagram



Technical Implementation

1. Used MySql and Python
2. Built a script in Sublime text editor, to build the connection between MySql and Python
3. Replicated all tables and data
4. Demonstrated CRUD operations

```
CA\Users\julia\Documents\ConnectionScript.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
ConnectionScript.py x test.py x
52
53 mycursor.execute("SELECT * FROM User")
54 print('Here are all the users:\n')
55 for x in mycursor:
56     print(x)

Here are all the users:


('AstuteWalrus', 'Homer', 'Pace', 'hpace@pace.edu', '19061906')
('Fluffybunny123', REDACTED, REDACTED, 'fluffy!!')
('FunnyCapybara', REDACTED, REDACTED, 'ovecapybara1234')
[Finished in 511ms]
```

```
CA\Users\julia\Documents\ConnectionScript.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
ConnectionScript.py x test.py x
57
58 mycursor.execute("INSERT INTO Product (product_type, product_name, brand, product_price) VALUES (%s, %s, %s, %s)",('Makeup', '
    Precisely, My Brow Wax', 'Benefit Cosmetics', 27.00))
59 mycursor.execute("INSERT INTO Product (product_type, product_name, brand, product_price) VALUES (%s, %s, %s, %s)",('Makeup', '
    Major Glow Balm', 'Patrick Ta Beauty', 50.00))
60 mycursor.execute("INSERT INTO Product (product_type, product_name, brand, product_price) VALUES (%s, %s, %s, %s)",('Skincare', '
    Vanilla Lip Care Duo', 'Laneige', 34.00))
61 mycursor.execute("INSERT INTO Product (product_type, product_name, brand, product_price) VALUES (%s, %s, %s, %s)",('Skincare', '
    The Silk Serum', 'Tatcha', 98.00))
62 db.commit()
63
64 mycursor.execute("SELECT * FROM Product")
65 print('Here are all the products:\n')
66 for x in mycursor:
67     print(x)

Here are all the products:

(1, 'Makeup', 'Precisely, My Brow Wax', 'Benefit Cosmetics', Decimal('27.00'))
(2, 'Makeup', 'Major Glow Balm', 'Patrick Ta Beauty', Decimal('50.00'))
(3, 'Skincare', 'Vanilla Lip Care Duo', 'Laneige', Decimal('34.00'))
(4, 'Skincare', 'The Silk Serum', 'Tatcha', Decimal('98.00'))
[Finished in 784ms]
```

Deliverable #3 - User Interface




WELCOME

Please sign in:

USERNAME:




PASSOWORD







HOME

Search for a product ...

Today's Hot Products:


	★★★★★
	★★★★★
	★★★★★











COOLING WATER JELLY


Milk Makeup




BRAND: MILK MAKEUP
PRICE: \$24

	★★★★★
	★★★★★
	★★★★★











MY PROFILE



Reviews

	★★★★★
	★★★★★
	★★★★★



Security Measures



CAPTCHA required
during login



Encryption of
passwords and emails



We will not sell your
data or track your
data to be sold!

More Complex Queries

1. Select all reviews for all products and sort in descending order by Overall_5rating.
2. Select and group products by product name, and show the average Overall_5rating for each product.

```
155 '''
156
157 #selects For instance, get all reviews for the product _ and sort in descending order.
158 try:
159     mycursor.execute("SELECT p.product_name, p.brand, p.product_price, r.short_response, r.overall_5rating FROM Product as p, Review as r WHERE p.
        product_ID = r.product_ID ORDER BY r.overall_5rating DESC")
160
161     print('Here are all the reviews:\n')
162
163     for x in mycursor:
164         print(x)
165 except:
166     print("Could not select data, please check your syntax.")
167
```

Here are all the reviews:

```
('Precisely, My Brow Wax', 'Benefit Cosmetics', Decimal('27.00'), 'Product keeps my eyebrows in place all day!', Decimal('5'))
('Precisely, My Brow Wax', 'Benefit Cosmetics', Decimal('27.00'), 'Product keeps my eyebrows in place all day!', Decimal('5'))
('The Silk Serum', 'Tatcha', Decimal('98.00'), 'This product is my holy grail', Decimal('5'))
('Vanilla Lip Care Duo', 'Laneige', Decimal('34.00'), 'Keeps my lips super smooth all day long.', Decimal('5'))
('Precisely, My Brow Wax', 'Benefit Cosmetics', Decimal('27.00'), 'I like this product a lot, just wish the packaging was less ugly.', Decimal('4'))
('Major Glow Balm', 'Patrick Ta Beauty', Decimal('50.00'), 'This product is a little out of my budget, but works very well!', Decimal('4'))
('Vanilla Lip Care Duo', 'Laneige', Decimal('34.00'), 'I hate the taste and smell of this product. Never buying it again', Decimal('2'))
('Major Glow Balm', 'Patrick Ta Beauty', Decimal('50.00'), 'This product is way too overpriced, did not work for my skin.', Decimal('1'))
[Finished in 588ms]
```

```
164     print(x)
165 except:
166     print("Could not select data, please check your syntax.")
167 '''
168
169 try:
170     mycursor.execute("SELECT p.product_name, AVG(r.overall_5rating) FROM Product as p, Review as r WHERE p.product_ID = r.product_ID GROUP BY p.
        product_name")
171
172     print('Here are all of the product names, each with the average 5-star rating for each product:\n')
173
174     for x in mycursor:
175         print(x)
176 except:
177     print("Could not select data, please check your syntax.")
178
```

Here are all of the product names, each with the average 5-star rating for each product:

```
('Precisely, My Brow Wax', Decimal('4.6667'))
('Major Glow Balm', Decimal('2.5000'))
('Vanilla Lip Care Duo', Decimal('3.5000'))
('The Silk Serum', Decimal('5.0000'))
[Finished in 546ms]
```

Enforcing a Unique Constraint

```
ConnectionScript.py  test.py x
181
182 mycursor.execute("DELETE FROM Review WHERE review_ID = 3")
183 db.commit()
184
185 mycursor.execute("SELECT * FROM Review")
186 for x in mycursor:
187     print(x)
188
189 #ensures that there is only one review, per person, per product
190 mycursor.execute("ALTER TABLE Review ADD CONSTRAINT OneReview_PerPerson_PerProduct UNIQUE (user_ID, product_ID)")
191
192 try:
193     mycursor.execute("INSERT INTO Review(user_ID, product_ID, short_response, effectiveness, value, phys_properties) VALUES (%s, %s, %s, %s, %s, %s)", ('Fluffybunny123',1,'Product
194     keeps my eyebrows in place all day!', 5, 5, 4))
195
196 except:
197     print("You have already made a review for this product! Please consider updating your review instead.")
198
(1, 'Fluffybunny123', 1, 'Product keeps my eyebrows in place all day!', 5, 5, 4, Decimal('5'))
(2, 'AstuteWalrus', 1, 'I like this product a lot, just wish the packaging was less ugly.', 5, 5, 2, Decimal('4'))
(4, 'FunnyCapybara', 2, 'This product is a little out of my budget, but works very well!', 5, 2, 4, Decimal('4'))
(5, 'Fluffybunny123', 4, 'This product is my holy grail', 5, 5, 5, Decimal('5'))
(6, 'AstuteWalrus', 3, 'Keeps my lips super smooth all day long.', 5, 5, 5, Decimal('5'))
(7, 'AstuteWalrus', 2, 'This product is way too overpriced, did not work for my skin.', 1, 1, 1, Decimal('1'))
(8, 'Fluffybunny123', 3, 'I hate the taste and smell of this product. Never buying it again', 1, 3, 1, Decimal('2'))
[Finished in 708ms]
```

```
ConnectionScript.py  x  test.py x
185
186 mycursor.execute("SELECT * FROM Review")
187 for x in mycursor:
188     print(x)
189 ...
190 #ensures that there is only one review, per person, per product
191 #mycursor.execute("ALTER TABLE Review ADD CONSTRAINT OneReview_PerPerson_PerProduct UNIQUE (user_ID, product_ID)")
192
193 #code to test unique constraint, tries to add an observation that violates the unique constraint
194 try:
195     mycursor.execute("INSERT INTO Review(user_ID, product_ID, short_response, effectiveness, value, phys_properties) VALUES (%s, %s, %s, %s, %s, %s)", ('Fluffybunny123',1,'Product
196     keeps my eyebrows in place all day!', 5, 5, 4))
197
198 except:
199     print("You have already made a review for this product! Please consider updating your review instead.")
200
You have already made a review for this product! Please consider updating your review instead.
[Finished in 591ms]
```





Performance Optimization

1

Indexing on frequently queried columns like user_Id and product_ID

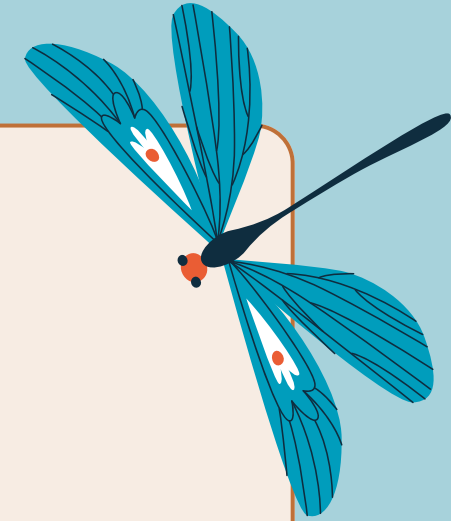
2

Read replicas in a distributed database for a read-heavy platform

3

Possible de-normalization if performance impact occurs from too many joins





THANK YOU!

Any questions?

