

Inventory Management and Online Order Tracking

Eva Pradhan
University at Buffalo
50414973
evapradh@buffalo.edu

Harshitha Damineni
University at Buffalo
50471119
hdaminen@buffalo.edu

Venkata Datta Viswanadha Vishnubhotla
University at Buffalo
50465091
vvishub@buffalo.edu

I. PROBLEM STATEMENT

A. Description

Most businesses need an inventory management database system to keep track of their products and stock levels, as well as an online order tracking database system to manage customer orders and shipments. If a system is not in place to keep track it can lead to overselling or stockouts, resulting in lost sales, missed sales opportunities and dissatisfied customers. Furthermore, without a system for tracking online orders, it can be challenging to monitor the status of orders, such as whether they have been fulfilled or shipped, causing delays in delivery times and customer dissatisfaction. Additionally, they require a user review system to allow customers to leave feedback on their purchases. It can be used for various analyses such as sales forecasting, customer segmentation, and product recommendation systems. On other hand, manual inventory tracking can be time-consuming and prone to errors, leading to inaccurate data and delays in decision making. Therefore, the need arises for an inventory management database system with an online order tracking database system that can efficiently manage and track small to large inventory levels, orders, and shipments. This system must be capable of monitoring real-time stock levels, generating alerts for low stock levels, managing purchase orders, and updating inventory levels automatically. Additionally, the system should be able to track online orders from the time they are placed to the point of delivery and provide customers with real-time updates on the status of their orders. With such a system in place, businesses can efficiently manage inventory levels, reduce the risk of stockouts, improve customer satisfaction, and streamline their overall operations. The system should be user-friendly, efficient, and scalable to accommodate the company's growth. The company also needs a robust reporting and analytics feature to monitor inventory levels, track inventory movements, and identify trends and patterns to make informed decisions. Finally, the system should have strong security features to protect sensitive inventory data from unauthorized access and cyber threats. The system should also allow customers to receive updates on their orders via email or SMS notifications. The system should be able to handle order information such as order status, delivery date, and shipping information. This will improve customer satisfaction by providing a seamless and transparent order tracking experience, which in turn will improve customer loyalty and increase sales for the business.

B. Database vs Excel Files

- In terms of data organization, consistency, security, scalability, and sharing/collaboration, databases outperform Excel files.
- While databases are intended to handle complicated queries and data interactions more effectively than Excel, they can provide greater performance when working with massive datasets. They also make it simple to create reports and visualizations from the data, enabling more insightful analysis and wiser decision-making.
- Moreover, databases give data a centralized location, lowering the possibility of data loss or corruption that can happen when Excel files are kept in many locations. This can reduce the time and effort required to retrieve lost or damaged data.
- Also, databases allow for more complex data conversions and calculations than Excel can, giving them greater versatility in terms of data manipulation. Businesses who need to run advanced analytics or machine learning on their data may find this to be extremely helpful.
- Overall, databases offer a more robust and scalable solution for enterprises with bigger volumes of data that require secure, consistent, and efficient management, whereas Excel can be effective for small datasets and straightforward analytics.

II. TARGET USER

Depending on the size and nature of the company, as well as the particular needs of the system, a variety of workforce may be the target users for an inventory management system and online order tracking. Here are a few examples of the intended users:

- E-commerce business owners: These people are in charge of overseeing the overall operations of the e-commerce business. They can use the management system to track sales, inventory levels, order management, and customer behavior.
- Customer service representatives: These individuals have the responsibility of responding to consumer questions and addressing any problems with orders. They can view order histories, access customer information, and handle returns or refunds via the management system.
- Logistics and shipping personnel: These people are responsible for organizing the shipping and delivery of

items to clients. Users could create shipping labels, track orders, and manage inventories using the management system.

- Inventory managers and supervisors: These users are in charge of monitoring inventory levels and ensuring that products are available when needed. An inventory management system can assist them in tracking inventory levels, monitoring sales, and making informed decisions about purchasing and stocking products.
- Purchasing and procurement staff: These users are in charge of ordering new inventory and ensuring that the company has the materials and products it needs to run efficiently. An inventory management system can help them track orders, manage suppliers, and ensure that the right products are available at the right time.
- Accounting and finance personnel: These people are in charge of overseeing the business's finances, which includes keeping track of inventory expenses and sales revenue.
- Sales representatives: These people are in charge of processing customer orders and making sure that orders are delivered promptly and accurately. They may create reports, manage inventories, and track orders using the management system.
- Data analysts: In order to find trends and patterns in customer behavior, product sales, and money generated by the e-commerce platform, these people may use the database management system to extract data from the system and conduct analysis.

III. REAL LIFE SCENARIO

A retail company that sells things both online and in-person might be a real-world example of how an inventory management system and online order monitoring are used. The company maintains a sizable inventory of goods that are kept in one or more warehouses. The products' quantities, locations, and any status changes would all be tracked in real-time by the inventory management system (such as being shipped or received).

The online order tracking system enables customers to follow their orders when they are placed online and up until they are delivered. The user would be able to view the pick-up, packing, and shipment dates of the purchase as well as any tracking details offered by the shipping carrier. As orders are filled, the inventory management system would automatically update the inventory levels.

A product's availability on the website will be automatically updated by the inventory management system whenever it runs out of stock. The technology would also send alerts when inventory levels reached a predetermined level, urging the company to place new orders for goods and replenish the warehouse.

The company can guarantee that it always has enough inventory to fulfill customer demand by putting in place an efficient inventory management system and an online order

tracking system, as well as by giving customers the ability to track their orders in real-time.

Here are some potential real-life scenarios where this dataset could be used:

- Market Analysis: The dataset can be used to assess market trends, including popular product categories, customer behavior, and seller performance, by an online retailer interested in breaking into the market. The corporation can use this research to make well-informed decisions regarding the products it offers and the way it markets those products.
- Customer Segmentation: An online store can use the dataset to classify its customers based on their demographics, buying habits, and other traits. The business may boost consumer engagement and loyalty by tailoring its marketing messaging and promotions to certain customer groups.
- Supply Chain Optimization: A logistics business might use the dataset to assess how successfully different suppliers manage orders and adhere to delivery dates. This analysis can help the company optimize its supply chain by identifying reliable providers and accelerating delivery.
- Fraud Detection: The dataset can be used by an online retailer to spot phony orders and sales. The business can create methods and algorithms to stop fraud in the future by studying the patterns of fraudulent activity.
- Product Recommendation: An online store can use the dataset to develop algorithms that recommend products to customers based on their prior purchases and browsing patterns. As a result, customers might be more inclined to make additional purchases, which will improve their whole shopping experience.

IV. DATA DESCRIPTION

The e-commerce market's dataset contains data on orders, consumers, sellers, products, and reviews. The e-commerce platform Olist, which offers marketplace services to small and medium-sized firms, is the source of the dataset.

The dataset is composed of multiple tables, including:

- Seller dataset: This file includes details about the marketplace vendors, including seller ID, seller name, and seller zip code.
- Customers dataset: This file includes details about the customers who placed the orders, including their name, city, state, and zip code, as well as a special customer ID.
- Order items dataset: Each order's items are listed in this file together with details such as the order ID, product ID, seller ID, price, freight value, and quantity.
- Geolocation dataset: The latitude and longitude of the buyers' and sellers' locations are included in this file along with other geolocation data.
- Order payments dataset: This file includes details on payments made for each order, including the value, kind, and installments of payments.

- Orders dataset: The order ID, customer ID, order status, purchase timestamp, and anticipated delivery date are just a few of the details regarding the orders that are included in this file.
- Products dataset: The product ID, product category name, product name, and product description are among the details about the goods offered on the marketplace that are included in this file.
- Order reviews dataset: The order ID, review score, review comment title, and review comment message are among the details regarding the reviews that consumers have written for the orders that are included in this file.
- Order items dataset: Each item of an order is described in detail in the order Items dataset. For each item, the dataset specifically offers details such as order id, order item id, product id, seller id, shipping limit date, etc.
- Product Seller dataset: This table maps the product id to corresponding seller id.

In conclusion, the E-Commerce Public Dataset by Olist offers a wealth of information for studying the e-commerce business, including consumer behavior, market trends, and seller performance.

V. ER DIAGRAM

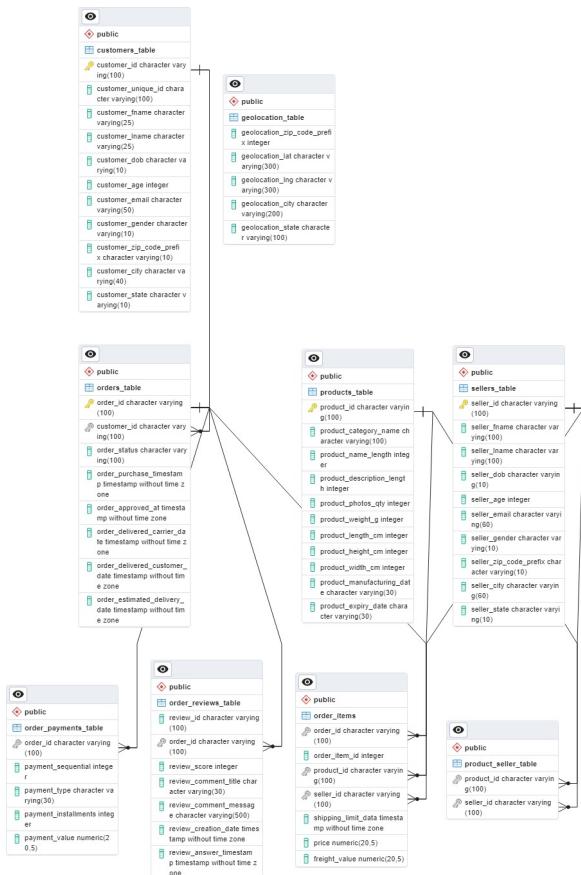


Fig. 1. ER diagram

VI. DATABASE SCHEMAS

- SELLERS_TABLE (seller_id VARCHAR primary key, seller_zip_code_prefix INT, seller_city VARCHAR, seller_state VARCHAR, seller_fname VARCHAR, seller_lname VARCHAR, seller_email VARCHAR, seller_gender VARCHAR, seller_dob VARCHAR, age INT)
- PRODUCTS_TABLE (product_id VARCHAR primary key, product_category_name VARCHAR, product_name_length INT, product_description_length INT, product_photos_qty INT, product_weight_g INT, product_length_cm INT, product_height_cm INT, product_width_cm INT, product_manufacturing_date VARCHAR, product_expiry_date VARCHAR)
- CUSTOMERS_TABLE (customer_id VARCHAR, customer_unique_id VARCHAR, customer_fname VARCHAR, customer_lname VARCHAR, customer_DOB VARCHAR, customer_age INT, customer_email VARCHAR, customer_zip_code_prefix VARCHAR, customer_gender VARCHAR, customer_city VARCHAR, customer_state VARCHAR)
- ORDERS_TABLE (order_id VARCHAR, customer_id, order_status VARCHAR, order_purchase_timestamp TIMESTAMP, order_approved_at TIMESTAMP, order_delivered_carrier_date TIMESTAMP, order_delivered_customer_date TIMESTAMP, order_estimated_delivery_date TIMESTAMP)
- ORDER_ITEMS_TABLE (order_item_id VARCHAR, order_id VARCHAR, product_id, seller_id, shipping_limit_date TIMESTAMP, price NUMERIC, freight_value NUMERIC)
- ORDER_PAYMENTS_TABLE (order_id VARCHAR, payment_sequential INT, payment_type VARCHAR, payment_installments INT, payment_value NUMERIC)
- ORDER_REVIEWS_TABLE (review_id VARCHAR, order_id VARCHAR, review_score INT, review_comment_title VARCHAR, review_comment_message VARCHAR, review_creation_date TIMESTAMP, review_answer_timestamp TIMESTAMP)
- GEOLOCATION_TABLE (geolocation_zip_code_prefix VARCHAR primary key, geolocation_lat VARCHAR, geolocation_lng VARCHAR, geolocation_city VARCHAR, geolocation_state VARCHAR)
- PRODUCT_SELLER_TABLE (product_id VARCHAR primary key, seller_id VARCHAR)

VII. DESCRIPTIONS OF RELATIONS AND THEIR ATTRIBUTES:

The following is a description of each relation and its attributes of the dataset we used for inventory management and online order tracking:

- CUSTOMERS_TABLE**: The customers who place orders on the e-commerce platform are represented by this relation. This table's attributes are as follows:

- customer_id: Each customer's unique identifier.
 - customer_unique_id: An identifier that is unique to each customer and remains the same across all orders.
 - customer_fname: The first name of the customer.
 - customer_lname: The last name of the customer.
 - customer_zip_code_prefix: The customer's location's zip code prefix.
 - customer_city: The city in which the customer resides.
 - customer_state: The state in which the customer resides.
 - customer_email: The email of the customer.
 - customer_DOB: The date of birth of the customer.
 - customer_age: The age of the customer.
 - customer_gender: The gender of the customer.
- ORDERS_TABLE: This relation represents customer orders placed on the e-commerce platform. This table's attributes are as follows:
 - order_id: The order's unique identifier.
 - customer_id: The unique identifier for the customer who made the purchase.
 - order_purchase_timestamp: The date and time of the order.
 - order_approved_at: The date and time that the e-commerce platform approved the order.
 - order_delivered_carrier_date: The date and time when the order was delivered to the carrier.
 - order_delivered_customer_date: The time and date the order was delivered to the customer.
 - order_estimated_delivery_date: The order's expected delivery date.
 - ORDER_ITEMS_TABLE: This relation represents the items in every customer order. This table's attributes are:
 - order_item_id: The order item's unique identifier.
 - order_id: The order to which the item belongs.
 - product_id: A unique identifier for each product.
 - seller_id: The unique identifier for the seller who sold the item.
 - shipping_limit_date: The deadline for the seller to ship the product.
 - price: The price of the product.
 - freight_value: The shipping cost of the product.
 - PRODUCTS_TABLE: This relation represents the e-commerce platform's products. This table's attributes are as follows:
 - product_id: Each product's unique identifier.
 - product_category_name: The category to which the product belongs.
 - product_name_length: The length of the product name.
 - product_description_length: The length of the product description.
 - product_manufacturing_date: The manufacturing date of the product.
 - product_expiry_date: The expiry date of the product.
 - product_photos_qty: The number of product photos available.
 - product_weight_g: The product's weight in grams.
 - product_length_cm: The product's length in centimeters.
 - product_height_cm: The product's height in centimeters.
 - product_width_cm: The product's width in centimeters.
 - SELLERS_TABLE: This relation represents the sellers who sell products on the e-commerce platform. This relation has the following attributes:
 - seller_id: A unique identifier for each seller.
 - seller_fname: The first name of the seller.
 - seller_lname: The last name of the seller.
 - seller_DOB: The date of birth of the seller.
 - seller_age: The age of the seller.
 - seller_zip_code: The zip code of the seller.
 - seller_zip_code_prefix: The seller's zip code prefix.
 - seller_city: The city in which the seller is located.
 - seller_state: The seller's home state.
 - seller_gender: The gender of the seller.
 - seller_email: The email of the seller.
 - ORDER_PAYMENTS_TABLE: This relation represents the payments made for each order. This relation has the following attributes:
 - order_id: The identifier for the order for which payment was made.
 - payment_sequential: The order's payment sequence number.
 - payment_type: The type of payment used.
 - payment_installments: The number of installments in which the payment was made.
 - payment_value: The payment's monetary value.
 - ORDER_REVIEWS_TABLE: This table contains information about customer reviews for orders they placed, as well as any responses from sellers. This table has the following attributes:
 - review_id: A unique identifier for the review
 - order_id: A unique identifier for the review order
 - review_score: The rating given by the customer in the review, ranging from 1 (worst) to 5 (best)
 - review_comment_title: The title of the review, which summarizes the customer's experience.
 - review_comment_message: The text of the review that describes the customer's experience.
 - review_creation_date: The date and time when the customer created the review
 - review_answer_timestamp: The date and time when the seller responded to the review, if applicable.
 - GEOLOCATION_TABLE: This table includes geographical information for each zip code prefix in an area, such as latitude and longitude coordinates, as well as the city and state. This table has the following attributes:

- geolocation_zip_code_prefix: The first five digits of the zip code that identify the region in which it is located.
 - geolocation_lat: The latitude of the zip code's associated location.
 - geolocation_lng: The longitude of the location associated with the zip code.
 - geolocation_city: The name of the city associated with the zip code.
 - geolocation_state: Abbreviation for the state associated with the zip code.
- PRODUCT_SELLER_TABLE: This table maps the product id to corresponding seller id
- product_id: product id of products
 - seller_id: seller id of corresponding sellers

VIII. RELATION BETWEEN TABLES

The relations between tables that exist in the dataset used for this system are:

- The ORDERS table has a one-to-many relationship with the ORDER_ITEMS table, as each order can have multiple items.
- The ORDER_ITEMS table has a many-to-one relationship with the ORDERS table, as each item can only belong to one order.
- The ORDER_ITEMS table has a many-to-one relationship with the PRODUCTS table, as each item is associated with one product.
- The ORDER_ITEMS table has a many-to-one relationship with the SELLERS table, as each item is sold by one seller.
- The SELLERS table has a one-to-many relationship with the ORDERS table, as each seller can have multiple orders.
- The CUSTOMERS table has a one-to-many relationship with the ORDERS table, as each customer can have multiple orders.
- The ORDER_REVIEWS table has a many-to-one relationship with the ORDERS table, as each review is associated with one order.
- The GEOLOCATION table is not directly related to any other table, but it can be used to obtain geographic information for orders and customers based on their zip codes.
- The PRODUCT_SELLER_TABLE has many to one mapping to the SELLERS table, as multiple products can belong to each seller

IX. TABLES AND KEYS

| Tables | Keys |
|----------------------|--|
| CUSTOMERS_TABLE | Primary key: customer_id Foreign key: customer_zip_code_prefix Foreign key references to GEOLOCATION_TABLE Not NULL: customer_unique_id, customer_fname, customer_DOB, customer_email, customer_gender |
| ORDERS_TABLE | Primary key: order_id Foreign key: customer_id Foreign key references to CUSTOMERS_TABLE Not NULL: order_status, order_purchase_timestamp |
| PRODUCTS_TABLE | Primary key: product_id |
| SELLERS_TABLE | Primary key: seller_id Foreign key: seller_zip_code_prefix Foreign key references to GEOLOCATION_TABLE Not NULL: seller_fname, seller_email, seller_gender, seller_dob |
| ORDER_PAYMENTS_TABLE | Foreign key: order_id Foreign key references to ORDERS_TABLE Not NULL: payment_sequential, payment_type, payment_installments, payment_value |
| ORDER_ITEMS_TABLE | Primary key: order_item_id Foreign key: order_id, product_id, seller_id Foreign key references to ORDERS_TABLE Foreign key references to PRODUCTS_TABLE Foreign key references to SELLERS_TABLE Not NULL: price |
| ORDER_REVIEWS_TABLE | Foreign key: order_id Foreign key references to ORDERS_TABLE Not NULL: review_score, review_creation_date |
| GEOLOCATION_TABLE | Primary key: geolocation_zip_code_prefix |
| PRODUCT_SELLER_TABLE | Foreign key: product_id Foreign key: seller_id |

X. FUNCTIONAL DEPENDENCIES

The functional dependencies for each table in the database schema are:

- SELLERS_TABLE:
 $\text{seller_id} \rightarrow \text{seller_fname}, \text{seller_lname}, \text{email}, \text{gender}, \text{date_of_birth}, \text{age}, \text{seller_zip_code_prefix}, \text{seller_city}, \text{seller_state}$
- PRODUCTS_TABLE:
 $\text{product_id} \rightarrow \text{product_category_name}, \text{product_name_length}, \text{product_description_length}, \text{product_manufacturing_date}, \text{product_expiry_date}, \text{product_photos_qty}, \text{product_weight_g}, \text{product_length_cm}, \text{product_height_cm}, \text{product_width_cm}, \text{manufacturing_date}, \text{expiry_date}$
- CUSTOMERS_TABLE:
 $\text{customer_id} \rightarrow \text{customer_unique_id}, \text{customer_fname}, \text{customer_lname}, \text{customer_DOB}, \text{customer_email}, \text{customer_zip_code_prefix}, \text{customer_gender}, \text{customer_age}, \text{customer_zip_code_prefix}, \text{customer_city}, \text{customer_state}$
- ORDERS_TABLE:
 $\text{order_id} \rightarrow \text{customer_id}, \text{order_status}, \text{order_purchase_timestamp}, \text{order_delivered_carrier_date}, \text{order_delivered_customer_date}, \text{order_estimated_delivery_date}$
- ORDER_ITEMS_TABLE:
 $\text{order_id} \rightarrow \text{order_item_id}, \text{product_id}, \text{seller_id}, \text{shipping_limit_date}, \text{price}, \text{freight_value}$

- ORDER_PAYMENTS_TABLE:
order_id → payment_sequential, payment_type, payment_installments, payment_value
- ORDER_REVIEWS_TABLE:
review_id → order_id, review_score, review_comment_title, review_comment_message, review_creation_date, review_answer_timestamp
- GEOLOCATION_TABLE:
geolocation_zip_code_prefix → geolocation_lat, geolocation_lng, geolocation_city, geolocation_state
- PRODUCT_SELLER_TABLE:
product_id → seller_id

The Functional Dependencies listed above for our relatives are all non-trivial FDs. Additionally, the super keys of that table are on the left side of each of these non-trivial functional dependencies. Therefore, we can state that BCNF is the relational schema used by all of the tables in our database.

XI. DATABASE IMPLEMENTATION

A. Creating Tables

```

57  create table sellers_table(seller_id VARCHAR(100) primary key,
58      seller_fname VARCHAR(100),
59      seller_lname VARCHAR(100),
60      seller_dob VARCHAR(10),
61      seller_age INT,
62      seller_email VARCHAR(60),
63      seller_gender VARCHAR(10),
64      seller_zip_code_prefix VARCHAR(10),
65      seller_city VARCHAR(60),
66      seller_state VARCHAR(10))
67
68
69  select count(*) from sellers table
Data Output Messages Notifications
CREATE TABLE
Query returned successfully in 58 msec.

```

Fig. 2. Creation of Sellers table

```

165  create table products_table(product_id VARCHAR(100) primary key,
166      product_category_name VARCHAR(100),
167      product_name_length INT,
168      product_description_length INT,
169      product_photos_qty INT,
170      product_weight_g INT,
171      product_length_cm INT,
172      product_height_cm INT,
173      product_width_cm INT,
174      product_manufacturing_date VARCHAR(30),
175      product_expiry_date VARCHAR(30))
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
7010
7011
7012
7013
7014
7015
7016
7017
7018
7019
7020
7021
7022
7023
7024
7025
7026
7027
7028
7029
70210
70211
70212
70213
70214
70215
70216
70217
70218
70219
70220
70221
70222
70223
70224
70225
70226
70227
70228
70229
70230
70231
70232
70233
70234
70235
70236
70237
70238
70239
70240
70241
70242
70243
70244
70245
70246
70247
70248
70249
70250
70251
70252
70253
70254
70255
70256
70257
70258
70259
702510
702511
702512
702513
702514
702515
702516
702517
702518
702519
702520
702521
702522
702523
702524
702525
702526
702527
702528
702529
702530
702531
702532
702533
702534
702535
702536
702537
702538
702539
702540
702541
702542
702543
702544
702545
702546
702547
702548
702549
702550
702551
702552
702553
702554
702555
702556
702557
702558
702559
702560
702561
702562
702563
702564
702565
702566
702567
702568
702569
702570
702571
702572
702573
702574
702575
702576
702577
702578
702579
702580
702581
702582
702583
702584
702585
702586
702587
702588
702589
7025810
7025811
7025812
7025813
7025814
7025815
7025816
7025817
7025818
7025819
7025820
7025821
7025822
7025823
7025824
7025825
7025826
7025827
7025828
7025829
70258210
70258211
70258212
70258213
70258214
70258215
70258216
70258217
70258218
70258219
70258220
70258221
70258222
70258223
70258224
70258225
70258226
70258227
70258228
70258229
70258230
70258231
70258232
70258233
70258234
70258235
70258236
70258237
70258238
70258239
70258240
70258241
70258242
70258243
70258244
70258245
70258246
70258247
70258248
70258249
70258250
70258251
70258252
70258253
70258254
70258255
70258256
70258257
70258258
70258259
70258260
70258261
70258262
70258263
70258264
70258265
70258266
70258267
70258268
70258269
70258270
70258271
70258272
70258273
70258274
70258275
70258276
70258277
70258278
70258279
70258280
70258281
70258282
70258283
70258284
70258285
70258286
70258287
70258288
70258289
70258290
70258291
70258292
70258293
70258294
70258295
70258296
70258297
70258298
70258299
702582100
702582101
702582102
702582103
702582104
702582105
702582106
702582107
702582108
702582109
702582110
702582111
702582112
702582113
702582114
702582115
702582116
702582117
702582118
702582119
702582120
702582121
702582122
702582123
702582124
702582125
702582126
702582127
702582128
702582129
702582130
702582131
702582132
702582133
702582134
702582135
702582136
702582137
702582138
702582139
702582140
702582141
702582142
702582143
702582144
702582145
702582146
702582147
702582148
702582149
702582150
702582151
702582152
702582153
702582154
702582155
702582156
702582157
702582158
702582159
702582160
702582161
702582162
702582163
702582164
702582165
702582166
702582167
702582168
702582169
702582170
702582171
702582172
702582173
702582174
702582175
702582176
702582177
702582178
702582179
702582180
702582181
702582182
702582183
702582184
702582185
702582186
702582187
702582188
702582189
702582190
702582191
702582192
702582193
702582194
702582195
702582196
702582197
702582198
702582199
702582200
702582201
702582202
702582203
702582204
702582205
702582206
702582207
702582208
702582209
702582210
702582211
702582212
702582213
702582214
702582215
702582216
702582217
702582218
702582219
702582220
702582221
702582222
702582223
702582224
702582225
702582226
702582227
702582228
702582229
702582230
702582231
702582232
702582233
702582234
702582235
702582236
702582237
702582238
702582239
702582240
702582241
702582242
702582243
702582244
702582245
702582246
702582247
702582248
702582249
702582250
702582251
702582252
702582253
702582254
702582255
702582256
702582257
702582258
702582259
702582260
702582261
702582262
702582263
702582264
702582265
702582266
702582267
702582268
702582269
702582270
702582271
702582272
702582273
702582274
702582275
702582276
702582277
702582278
702582279
702582280
702582281
702582282
702582283
702582284
702582285
702582286
702582287
702582288
702582289
702582290
702582291
702582292
702582293
702582294
702582295
702582296
702582297
702582298
702582299
702582300
702582301
702582302
702582303
702582304
702582305
702582306
702582307
702582308
702582309
702582310
702582311
702582312
702582313
702582314
702582315
702582316
702582317
702582318
702582319
702582320
702582321
702582322
702582323
702582324
702582325
702582326
702582327
702582328
702582329
702582330
702582331
702582332
702582333
702582334
702582335
702582336
702582337
702582338
702582339
702582340
702582341
702582342
702582343
702582344
702582345
702582346
702582347
702582348
702582349
702582350
702582351
702582352
702582353
702582354
702582355
702582356
702582357
702582358
702582359
702582360
702582361
702582362
702582363
702582364
702582365
702582366
702582367
702582368
702582369
702582370
702582371
702582372
702582373
702582374
702582375
702582376
702582377
702582378
702582379
702582380
702582381
702582382
702582383
702582384
702582385
702582386
702582387
702582388
702582389
702582390
702582391
702582392
702582393
702582394
702582395
702582396
702582397
702582398
702582399
702582400
702582401
702582402
702582403
702582404
702582405
702582406
702582407
702582408
702582409
702582410
702582411
702582412
702582413
702582414
702582415
702582416
702582417
702582418
702582419
702582420
702582421
702582422
702582423
702582424
702582425
702582426
702582427
702582428
702582429
702582430
702582431
702582432
702582433
702582434
702582435
702582436
702582437
702582438
702582439
702582440
702582441
702582442
702582443
702582444
702582445
702582446
702582447
702582448
702582449
702582450
702582451
702582452
702582453
702582454
702582455
702582456
702582457
702582458
702582459
702582460
702582461
702582462
702582463
702582464
702582465
702582466
702582467
702582468
702582469
702582470
702582471
702582472
702582473
702582474
702582475
702582476
702582477
702582478
702582479
702582480
702582481
702582482
702582483
702582484
702582485
702582486
702582487
702582488
702582489
702582490
702582491
702582492
702582493
702582494
702582495
702582496
702582497
702582498
702582499
702582500
702582501
702582502
702582503
702582504
702582505
702582506
702582507
702582508
702582509
702582510
702582511
702582512
702582513
702582514
702582515
702582516
702582517
702582518
702582519
702582520
702582521
702582522
702582523
702582524
702582525
702582526
702582527
702582528
702582529
702582530
702582531
702582532
702582533
702582534
702582535
702582536
702582537
702582538
702582539
702582540
702582541
702582542
702582543
702582544
702582545
702582546
702582547
702582548
702582549
702582550
702582551
702582552
702582553
702582554
702582555
702582556
702582557
702582558
702582559
702582560
702582561
702582562
702582563
702582564
702582565
702582566
702582567
702582568
702582569
702582570
702582571
702582572
702582573
702582574
702582575
702582576
702582577
702582578
702582579
702582580
702582581
702582582
702582583
702582584
702582585
702582586
702582587
702582588
702582589
702582590
702582591
702582592
702582593
702582594
702582595
702582596
702582597
702582598
702582599
702582600
702582601
702582602
702582603
702582604
702582605
702582606
702582607
702582608
702582609
702582610
702582611
702582612
702582613
702582614
702582615
702582616
702582617
702582618
702582619
702582620
702582621
702582622
702582623
702582624
702582625
702582626
702582627
702582628
702582629
702582630
702582631
702582632
702582633
702582634
702582635
702582636
702582637
702582638
702582639
702582640
702582641
702582642
702582643
702582644
702582645
702582646
702582647
702582648
702582649
702582650
702582651
702582652
702582653
702582654
702582655
702582656
702582657
702582658
702582659
702582660
702582661
702582662
702582663
702582664
702582665
702582666
702582667
702582668
702582669
702582670
702582671
702582672
702582673
702582674
702582675
702582676
702582677
702582678
702582679
702582680
702582681
702582682
702582683
702582684
702582685
702582686
702582687
702582688
702582689
702582690
702582691
702582692
702582693
702582694
702582695
702582696
702582697
702582698
702582699
702582700
702582701
702582702
702582703
702582704
702582705
702582706
702582707
702582708
702582709
702582710
702582711
702582712
702582713
702582714
702582715
702582716
702582717
702582718
702582719
702582720
702582721
702582722
702582723
702582724
702582725
702582726
702582727
702582728
702582729
702582730
702582731
702582732
702582733
702582734
702582735
702582736
702582737
702582738
702582739
702582740
702582741
702582742
702582743
702582744
702582745
702582746
702582747
702582748
702582749
702582750
702582751
702582752
702582753
702582754
702582755
702582756
702582757
702582758
702582759
702582760
702582761
70258276
```

```

129 create table order_reviews_table(review_id VARCHAR(100),
130         order_id VARCHAR(100),
131         review_score INT,
132         review_comment_title VARCHAR(30),
133         review_comment_message VARCHAR(500),
134         review_creation_date TIMESTAMP,
135         review_answer_timestamp TIMESTAMP,
136         FOREIGN KEY (order_id) REFERENCES orders_table(order_id))
137

```

CREATE TABLE

Query returned successfully in 77 msec.

Fig. 7. Creation of order_reviews table

```

218 create table product_seller_table(product_id VARCHAR(100) NOT NULL,
219         seller_id VARCHAR(100) NOT NULL,
220         FOREIGN KEY (seller_id) REFERENCES sellers_table(seller_id),
221         FOREIGN KEY (product_id) REFERENCES products_table(product_id))
222

```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 53 msec.

Fig. 10. Creation of product_sellers table

```

110 create table order_payments_table(order_id VARCHAR(100),
111         payment_sequential INT,
112         payment_type VARCHAR(30),
113         payment_installments INT,
114         payment_value NUMERIC(20,5),
115         FOREIGN KEY (order_id) REFERENCES orders_table(order_id))
116

```

CREATE TABLE

Query returned successfully in 49 msec.

Fig. 8. Creation of order_payments table

B. Loading data to PgAdmin4

For loading from csv files to the PostgreSQL for each dataset we have created table for a well-defined structure to store the CSV file data and imported the CSV file of that dataset containing data into PostgreSQL.

For creating the table structure for each table we have executed the above mentioned create queries for the tables.

For importing the csv files to PostgreSQL for all the tables we have imported as shown in the figure below.

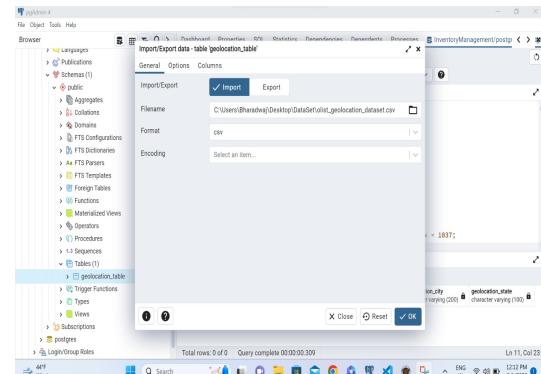


Fig. 11. Importing a table's csv file to PostgreSQL

```

13 create table geolocation_table(geolocation_zip_code_prefix INT,
14         geolocation_lat VARCHAR(300),
15         geolocation_lng VARCHAR(300),
16         geolocation_city VARCHAR(200),
17         geolocation_state VARCHAR(100))
18

```

CREATE TABLE

Query returned successfully in 54 msec.

Fig. 9. Creation of geolocation table

C. Insertion

```

71 INSERT INTO sellers_table(seller_id,
72         seller_name,
73         seller_email,
74         seller_dob,
75         seller_age,
76         seller_gender,
77         seller_zip_code_prefix,
78         seller_city,
79         seller_state)
80 VALUES (35436, 'johny', 'depp', '04/21/96', '89', 'johny@gmail.com', 'male', '9999', 'MUMBAI', 'MAHARASHTRA')
81
82 SELECT * FROM sellers_table where seller_city = 'MUMBAI';
83

```

Data Output Messages Notifications

| | seller_id | seller_name | seller_email | seller_dob | seller_age | seller_gender | seller_zip_code_prefix |
|---|-----------|-------------|--------------|------------|------------|-----------------|------------------------|
| 1 | 35436 | johny | depp | 04/21/96 | 89 | johny@gmail.com | male |

Fig. 12. Inserting a record into Sellers table

```

78 INSERT INTO products_table(product_id,
79     product_category_name,
80     product_name_length,
81     product_description_length,
82     product_photos_qty,
83     product_weight_g,
84     product_length_cm,
85     product_height_cm,
86     product_width_cm,
87     product_manufacturing_date,
88     product_expiry_date)
89 VALUES(76979, 'dairy_milk', 10, 346, 2, 1000, 9, '2023-01-11', '2024-01-11')
90
91 Select * FROM products_table where product_id = 76979

```

Data Output Messages Notifications

| product_id | product_category_name | product_name_length | product_description_length | product_photos_qty | product_weight_g | product_length_cm | product_height_cm | product_width_cm |
|------------|-----------------------|---------------------|----------------------------|--------------------|------------------|-------------------|-------------------|------------------|
| 76979 | dairy_milk | 10 | 346 | 2 | 1000 | 9 | | |

Fig. 13. Inserting a record into products table

```

96 INSERT INTO order_items(order_id,
97     order_item_id,
98     product_id,
99     seller_id,
100    shipping_init_data,
101    price,
102    freight_value)
103 VALUES(4764, 76979, 76979, 35436, '2023-06-05 23:57', 99, 100)
104
105
106 Select * FROM products_table where product_id = 76979
107

```

Data Output Messages Notifications

| order_id | product_id | product_name_length | product_description_length | product_photos_qty | product_weight_g | product_length_cm | product_height_cm | product_width_cm |
|----------|------------|---------------------|----------------------------|--------------------|------------------|-------------------|-------------------|------------------|
| 1 | 76979 | dairy_milk | 10 | 346 | 2 | 1000 | 9 | |

Fig. 16. Inserting a record into order_items table

```

1124324
34205544
Stephen
Hawking
1928-07-08
897
steph@gmail.com
1234567890
MUMBAI

```

Fig. 14. Inserting a record into Customers table

```

146 INSERT INTO order_reviews_table(review_id,
147     order_id,
148     review_score,
149     review_comment_title,
150     review_comment_message,
151     review_creation_date,
152     review_answer_timestamp)
153 VALUES(3424, 4764, 4, 'Good', 'Very good', '2023-06-04 18:00', '2023-06-04 18:00')
154
155
156 Select * FROM order_reviews_table where review_id = 3424
157

```

Data Output Messages Notifications

| review_id | order_id | review_score | review_comment_title | review_comment_message | review_creation_date | review_answer_timestamp |
|-----------|----------|--------------|----------------------|------------------------|----------------------|-------------------------|
| 1 | 3424 | 4764 | 4 | Good | Very good | 2023-06-04 10:00:00 |

Fig. 17. Inserting a record into order_reviews table

```

4764
1124324
PLACED
2023-11-23 16:15:32
2023-11-29 16:10:00
2023-11-29 16:10:00
2023-11-29 16:10:00

```

Fig. 15. Inserting a record into Orders table

```

120 INSERT INTO order_payments_table(order_id,
121     payment_sequential,
122     payment_type,
123     payment_installments,
124     payment_value)
125 VALUES('4764', 2, 'credit_card', 99)
126
127 Select * FROM order_payments_table where order_id = '4764';

```

Data Output Messages Notifications

| order_id | payment_sequential | payment_type | payment_installments | payment_value |
|----------|--------------------|--------------|----------------------|---------------|
| 1 | 4764 | 2 | credit_card | 90.00000 |

Fig. 18. Inserting a record into order_payments table

```

1 INSERT INTO geolocation_table(geolocation_zip_code_prefix,
2                               geolocation_lat,
3                               geolocation_lng,
4                               geolocation_city,
5                               geolocation_state) VALUES(1234, 90, 112, 'MUMBAI', 'MAHARASHTRA')
6
7 SELECT * FROM geolocation_table WHERE geolocation_city = 'MUMBAI';
8
9 CREATE TABLE customers_table(id INT PRIMARY KEY
10                             );

```

Data Output Messages Notifications

| geolocation_zip_code_prefix | geolocation_lat | geolocation_lng | geolocation_city | geolocation_state |
|-----------------------------|-----------------|-----------------|------------------|-------------------|
| 1234 | 90 | 112 | MUMBAI | MAHARASHTRA |

Fig. 19. Inserting a record into geolocation table

```

320 UPDATE products_table
321 SET product_category_name = 'construcao_ferramentas_seguranca'
322 WHERE product_id = '96bd76ec8810374ed1b65e291975717f';

```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 79 msec.

Fig. 22. Update command on products table

```

124 INSERT INTO product_seller_table(product_id,
125                               seller_id)
126 VALUES(16979, 35436);
127
128 SELECT * FROM products_table WHERE product_id = '16979';
129

```

Data Output Messages Notifications

| product_id | seller_id |
|------------|-----------|
| 16979 | 35436 |

Fig. 20. Inserting a record into product_seller table

```

316 UPDATE customers_table
317 SET customer_city = 'New City'
318 WHERE customer_id = '9fb35e4ed6f0a14a4977cd9aea4042bb';

```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 134 msec.

Fig. 23. Update command on customers table

D. Updating tables

```

324 UPDATE sellers_table
325 SET seller_zip_code_prefix = '11111'
326 WHERE seller_id = 'c0f3eea2e14555b6faeea3dd58c1b1c3';

```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 163 msec.

Fig. 21. Update command on sellers table

```

307 UPDATE orders_table
308 SET order_status = 'delivered'
309 WHERE order_id = 'e481f51cbdc54678b7cc49136f2d6af7';

```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 84 msec.

Fig. 24. Update command on orders table

```

311 UPDATE order_items
312 SET price = 50.99
313 WHERE order_id = '000576fe39319847ccb9d288c5617fa6'
314 AND product_id = '557d850972a7d6f792fd18ae1400d9b6';

```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 135 msec.

Fig. 25. Update command on order_items table

E. Deletion

```

267
268
269
270 -- select * from geolocation_table;
271
272 -- insert into geolocation_table VALUES(1, 2, 3, 'new city', 'NC');
273
274 -- select * from geolocation_table where geolocation_zip_code_prefix = 1;
275
276 delete from geolocation_table where geolocation_zip_code_prefix = 1;
277
278
279
280
281
282

```

Data Output Messages Notifications

DELETE 1

Query returned successfully in 174 msec.

Fig. 26. Delete command on table

```

294
295
296
297 -- select seller_total_sales('');
298
299
300
301
302
303 -- select * from products_table where product_category_name = 'artes';
304
305 delete from products_table where product_category_name = 'artes';
306
307
308
309
310
311

```

Data Output Messages Notifications

DELETE 55

Query returned successfully in 200 msec.

Fig. 27. Delete command on table

F. SQL Queries

- 1) Query to find the top 10 customers who have made the highest number of purchases:
- 2) Query to find the average rating for each product category
- 3) Query to find the number of unique customers who have made a purchase in each month of 2018

```

230 SELECT customers.customer_unique_id,
231 COUNT(*) AS total_purchases
232 FROM customers_table AS customers
233 JOIN orders_table AS orders ON customers.customer_id = orders.customer_id
234 GROUP BY customers.customer_unique_id
235 ORDER BY total_purchases DESC
236 LIMIT 10;

```

Data Output Messages Notifications

| customer_unique_id | total_purchases |
|-----------------------------------|-----------------|
| 8d50fSeadfs0201ccdedfb9e2ac8455 | 17 |
| 3e43e6105506432c953e165fb2acf44c | 9 |
| ca77025e7201e3b30c44472f1f3462... | 7 |
| 1b6c7548a2a1f9037c1fd3ddfed95f33 | 7 |
| 646999c91f9dfa7733b25662ef7f1782 | 7 |
| 63fc61cee11cbe306bfff5857d00bfe4 | 6 |
| 12f5de1ecbf93da9dc19095df0b3d | 6 |
| f0e310a6839dce9dfe630e0fe5ab282a | 6 |
| d34b16117594161aa8a9c50b289d... | 6 |

Total rows: 10 of 10 Query complete 00:00:00.294

Fig. 28. Select query 1

```

38 SELECT product_category_name,
39 AVG(order_reviews.review_score) AS avg_rating
40 FROM products_table AS products
41 JOIN order_items AS order_items ON products.product_id = order_items.product_id
42 JOIN orders_table AS orders ON order_items.order_id = orders.order_id
43 JOIN order_reviews_table AS order_reviews ON orders.order_id = order_reviews.order_id
44 GROUP BY product_category_name;

```

Data Output Messages Notifications

| product_category_name | avg_rating |
|---------------------------|--------------------|
| agro_industria_e_comercio | 4.000000000000000 |
| alimentos | 4.2181818181818182 |
| alimentos_bebidas | 4.3154121863799283 |
| artes | 3.937190676328502 |
| artes_e_artesanato | 4.125000000000000 |
| artigos_de_festas | 3.7674418604651163 |
| artigos_de_natal | 4.0205479452054795 |
| audio | 3.8254847645429363 |

Total rows: 75 of 75 Query complete 00:00:00.254

Fig. 29. Select query 2

```

255 SELECT DATE_TRUNC('month', orders.order_purchase_timestamp) AS month,
256 COUNT(DISTINCT orders.customer_id) AS unique_customers
257 FROM orders_table AS orders
258 WHERE EXTRACT(YEAR FROM orders.order_purchase_timestamp) = 2018
259 GROUP BY month;

```

Data Output Messages Notifications

| month | unique_customers |
|---------------------|------------------|
| 2018-01-01 00:00:00 | 7269 |
| 2018-02-01 00:00:00 | 6728 |
| 2018-03-01 00:00:00 | 7211 |
| 2018-04-01 00:00:00 | 6939 |
| 2018-05-01 00:00:00 | 6873 |
| 2018-06-01 00:00:00 | 6167 |
| 2018-07-01 00:00:00 | 6292 |
| 2018-08-01 00:00:00 | 6512 |
| 2018-09-01 00:00:00 | 16 |
| 2018-10-01 00:00:00 | 4 |

Total rows: 10 of 10 Query complete 00:00:00.551

Fig. 30. Select query 3

```

51 SELECT customers.customer_city,
52     AVG(order_items.price + order_items.freight_value) AS avg_order_value
53     FROM customers_table AS customers
54 JOIN orders_table AS orders ON customers.customer_id = orders.customer_id
55 JOIN order_items AS order_items ON orders.order_id = order_items.order_id
56 GROUP BY customers.customer_city
57 ORDER BY avg_order_value DESC
58 LIMIT 10;

```

Data Output Messages Notifications

| customer_city | avg_order_value |
|--------------------------|---------------------------|
| pianco | 2324.99000000000000000000 |
| nova esperanca do piraia | 2252.66000000000000000000 |
| engenheiro navarro | 2106.55000000000000000000 |
| agrestina | 2066.34000000000000000000 |
| mariental | 1867.85000000000000000000 |
| loredo | 1643.64000000000000000000 |
| ibitita | 1534.58000000000000000000 |
| pirpirituba | 1372.25000000000000000000 |

Total rows: 10 of 10 Query complete 00:00:00.260

Fig. 31. Select query 4

- 4) Query to find the top 10 cities with the highest average order value
- 5) Query to find the top 5 products that have been returned the most:

```

70 SELECT order_items.product_id, products.product_category_name,
71     COUNT(*) AS return_count
72     FROM order_items AS order_items
73 JOIN orders_table AS orders ON order_items.order_id = orders.order_id
74 JOIN order_reviews_table AS order_reviews ON orders.order_id = order_reviews.order_id
75 JOIN products_table AS products ON order_items.product_id = products.product_id
76 WHERE order_reviews.review_comment_title LIKE '%devolução%'
77 GROUP BY order_items.product_id, products.product_category_name
78 ORDER BY return_count DESC
79 LIMIT 5;

```

Data Output Messages Notifications

| product_id | product_category_name | return_count |
|------------------------------------|------------------------|--------------|
| 89b121bee266ddc25688a1ba72eebf61 | informatica_acessorios | 2 |
| ea36811d0335dfdf45b7f1f1145a5562c7 | informatica_acessorios | 2 |
| 0798195b28e04ce8e54322299e8829.. | informatica_acessorios | 1 |
| 2b4042dfab3fe9d3f16951a8e1a370 | utilidades_domesticas | 1 |
| c1e8014cae9330629b1de89dfa5a1bb | relógios_presentes | 1 |

Total rows: 5 of 5 Query complete 00:00:00.125

Fig. 32. Select query 5

- 6) Query to find the average time it takes for a customer to receive their order after it has been shipped

```

272 SELECT AVG(EXTRACT(EPOCH FROM
273     (orders.order_delivered_customer_date - orders.order_delivered_carrier_date))) / 86400 AS avg_delivery_time_days
274     FROM orders_table AS orders
275 WHERE
276     orders.order_status = 'delivered' AND
277     orders.order_delivered_customer_date IS NOT NULL AND
278     orders.order_delivered_carrier_date IS NOT NULL

```

Data Output Messages Notifications

| avg_delivery_time_days | |
|------------------------|--|
| 9.3302970579831249 | |

Fig. 33. Select query 6

- 7) Query to find the top 10 categories that have the highest number of orders:
- 8) Query to find top 5 sellers who have the highest percentage of orders delivered on time (i.e., the order was delivered on or before the estimated delivery date)
- 9) Query to find the top 10 customers who have the highest average rating for products they have reviewed, along with the number of reviews they have written.

```

80 SELECT products.product_category_name,
81     COUNT(DISTINCT orders.order_id) AS order_count
82     FROM orders_table AS orders
83 JOIN order_items AS order_items ON orders.order_id = order_items.order_id
84 JOIN products_table AS products ON order_items.product_id = products.product_id
85 GROUP BY products.product_category_name
86 ORDER BY order_count DESC
87 LIMIT 10;

```

Data Output Messages Notifications

| product_category_name | order_count |
|------------------------|-------------|
| cama_mesa_banho | 9417 |
| beleza_saude | 8836 |
| esporte_lazer | 7720 |
| informatica_acessorios | 6689 |
| moveis_decoracao | 6449 |
| utilidades_domesticas | 5884 |
| relógios_presentes | 5624 |
| telefonia | 4199 |

Total rows: 10 of 10 Query complete 00:00:01.433

Fig. 34. Select query 7

```

SELECT s.seller_id,
    COUNT(CASE WHEN o.order_delivered_customer_date <= o.order_estimated_delivery_date THEN 1 END) /
    COUNT(*) AS on_time_delivery_rate
FROM sellers_table AS s
JOIN order_items AS oi ON s.seller_id = oi.seller_id
JOIN orders_table AS o ON oi.order_id = o.order_id
GROUP BY s.seller_id
ORDER BY on_time_delivery_rate DESC
LIMIT 5;

```

Data Output Messages Notifications

| seller_id | on_time_delivery_rate |
|----------------------------------|-----------------------|
| fc3885dceee17a30fad88343437fb | 1 |
| c53bcd3d9e457a42497e939a59f09e22 | 1 |
| 642f2e3e539480db1e042493a4e52e.. | 1 |
| a61ca0479330895a8408071043651.. | 1 |
| 5def4c3732941a971cba8fdee992ed1 | 1 |

Fig. 35. Select query 8

```

19 SELECT r.order_id,
20     COUNT(r.review_id) AS num_reviews,
21     AVG(r.review_score) AS avg_rating
22     FROM order_reviews_table AS r
23 GROUP BY r.order_id
24 ORDER BY avg_rating DESC
25 LIMIT 10;

```

Data Output Messages Notifications

| order_id | num_reviews | avg_rating |
|----------------------------------|-------------|------------------------------------|
| 0006ec9db01a64e59a68b2c340bf65a7 | 1 | 5.00000000000000000000000000000000 |
| 0009792311464db532ff765bf7b182ae | 1 | 5.00000000000000000000000000000000 |
| 00061f2a7bc09da83e415a52dc8a4af1 | 1 | 5.00000000000000000000000000000000 |
| 00063b381e2406b52ad429470734ebd5 | 1 | 5.00000000000000000000000000000000 |
| 000229ec398224ef6ca0657da4fc703e | 1 | 5.00000000000000000000000000000000 |
| 0008288aa423d2a3f00fb17cd7d8719 | 1 | 5.00000000000000000000000000000000 |
| 000576fe39319847ccb9d288c5617fa6 | 1 | 5.00000000000000000000000000000000 |
| 00010242fe8c5a6d1ba2d792cb16214 | 1 | 5.00000000000000000000000000000000 |
| 00042b26cf59d7ce69dfabb4e55b4fd9 | 1 | 5.00000000000000000000000000000000 |

Total rows: 10 of 10 Query complete 00:00:01.297

Fig. 36. Select query 9

XII. QUERY EXECUTION ANALYSES

A. TRIGGER COMMAND

In some cases when one table in the database is updated with insert, update or delete command we also need to update a related table. In such cases trigger command with function implementation helps carry out the changes in related tables. In our case, when a successful payment is made, the order_payments_table has a record inserted, which upon insertion triggers and update on the order_status column of orders_table

```

168
169 -- select * from order_payments_table where order_id = '476478';
170 select * from orders_table where order_id = '4764789';
171
172
173 -- INSERT INTO order_payments_table(order_id,
174 --                                     payment_sequential,
175 --                                     payment_type,
176 --                                     payment_installments,
177 --                                     payment_value)
178
179 -- VALUES('47648', 2, 'credit_card', 3, 90);
180
181
182
183
184
185

```

| order_id | customer_id | order_status | order_purchase_timestamp |
|----------|-------------|--------------|--------------------------|
| 4764789 | 1124324 | PLACED | 2016-12-23 10:57:32 |

Fig. 37. Trigger Command

```

1 create or replace function payment_made_function() returns trigger as $payment_mades
2 begin
3     if new.payment_installments > 0
4     then
5         update orders_table
6         set order_status = 'purchased'
7         from orders_table as ordernumber
8         where ordernumber.order_id = new.order_id;
9     end if;
10    return new;
11 end;
12 $payment_made$ LANGUAGE plpgsql;
13
14 create trigger payment_made
15 after INSERT
16 on
17 order_payments_table
18 for each row
19 execute function payment_made_function();
20
21
22

```

CREATE TRIGGER
Query returned successfully in 47 msec.

Fig. 38. Trigger Command

```

168
169 -- select * from order_payments_table where order_id = '4764789';
170 -- select * from orders_table where order_id = '4764789';
171
172
173 INSERT INTO order_payments_table(order_id,
174                                     payment_sequential,
175                                     payment_type,
176                                     payment_installments,
177                                     payment_value)
178
179 VALUES('4764789', 2, 'credit_card', 3, 90);
180
181
182
183
184
185

```

INSERT 0 1
Query returned successfully in 1 secs 221 msec.

Fig. 39. Trigger Command

```

168
169 -- select * from order_payments_table where order_id = '476478';
170 select * from orders_table where order_id = '4764789';
171
172
173 -- INSERT INTO order_payments_table(order_id,
174 --                                     payment_sequential,
175 --                                     payment_type,
176 --                                     payment_installments,
177 --                                     payment_value)
178
179 -- VALUES('47648', 2, 'credit_card', 3, 90);
180
181
182
183
184
185

```

Data Output

| order_id | customer_id | order_status | order_purchase_timestamp |
|----------|-------------|--------------|--------------------------|
| 4764789 | 1124324 | PLACED | 2016-12-23 10:57:32 |

Total rows: 1 of 1 Query complete 00:00:00.130

Fig. 40. Trigger Command

B. INDEXING

While running some complex select queries involving one or more databases with huge amount of data, we noticed a larger amount of time for query execution. In the example given, we run a select query involving a join operation on orders_table and customers_table. Upon adding an index to order_id in orders_table we see a reduction in the query execution time. Upon adding an index to customer_id in customers_table also we see a further reduction in the query execution time.

```

184
185 -- create index idx_orders_customer_id on orders_table(customer_id);
186
187 -- drop index idx_customers_customer_id;
188
189 -- drop index idx_orders_customer_id;
190
191 SELECT customers.customer_unique_id,
192     COUNT(*) AS total_purchases
193     FROM customers_table AS customers
194     JOIN orders_table AS orders ON customers.customer_id = orders.customer_id
195     GROUP BY customers.customer_unique_id
196     ORDER BY total_purchases DESC
197     LIMIT 10;
198

```

Data Output

Successfully run. Total query runtime: 254 msec.
10 rows affected.

Total rows: 10 of 10 Query complete 00:00:00.254

Fig. 41. Indexing

```

179 -- GROUP BY customers.customer_unique_id
180 -- ORDER BY total_purchases DESC
181 -- LIMIT 10;
182
183 create index idx_customers_customer_id on customers_table(customer_id);
184
185 -- create index idx_orders_customer_id on orders_table(order_id);
186
187 -- drop index idx_customers_customer_id;
188
189 -- drop index idx_orders_customer_id;
190
191 -- SELECT customers.customer_unique_id,
192 -- COUNT(*) AS total_purchases
193 -- FROM customers_table AS customers
194 -- JOIN orders_table AS orders ON customers.customer_id = orders.customer_id
195 -- ORDER BY total_purchases DESC
196 -- LIMIT 10;
197

```

Data Output

CREATE INDEX

Query returned successfully in 213 msec.

Total rows: 10 of 10 Query complete 00:00:00.213

✓ Query returned successfully

Fig. 42. Indexing

```

184
185 -- create index idx_orders_customer_id on orders_table(customer_id);
186
187 -- drop index idx_customers_customer_id;
188
189 -- drop index idx_orders_customer_id;
190
191 SELECT customers.customer_unique_id,
192     COUNT(*) AS total_purchases
193     FROM customers_table AS customers
194     JOIN orders_table AS orders ON customers.customer_id = orders.customer_id
195     GROUP BY customers.customer_unique_id
196     ORDER BY total_purchases DESC
197     LIMIT 10;
198

```

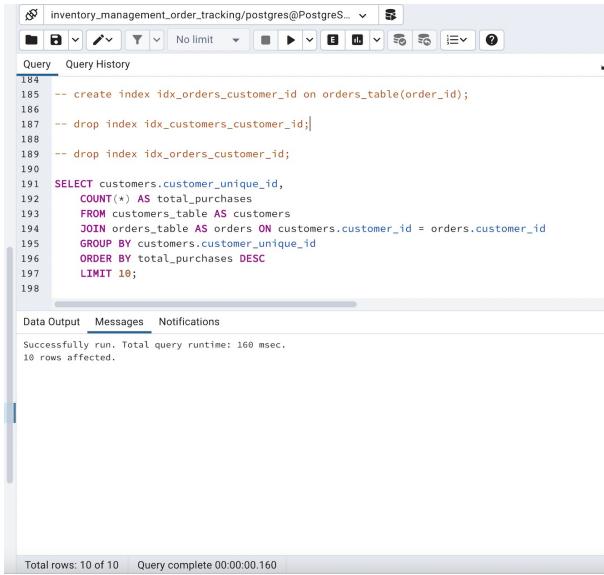
Data Output

Successfully run. Total query runtime: 254 msec.
10 rows affected.

Total rows: 10 of 10 Query complete 00:00:00.254

Fig. 43. Indexing

XIII. GUI



```

184 -- create index idx_orders_customer_id on orders_table(order_id);
185
186 -- drop index idx_customers_customer_id;
187
188 -- drop index idx_orders_customer_id;
189
190
191 SELECT customers.customer_unique_id,
192     COUNT(*) AS total_purchases
193     FROM customers_table AS customers
194     JOIN orders_table AS orders ON customers.customer_id = orders.customer_id
195     GROUP BY customers.customer_unique_id
196     ORDER BY total_purchases DESC
197     LIMIT 10;
198

```

Data Output Messages Notifications

Successfully run. Total query runtime: 160 msec.
10 rows affected.

Total rows: 10 of 10 Query complete 00:00:00.160

Fig. 44. Indexing

This application has been developed using JSP(Jakarta Server Pages) for the frontend and Java for the backend using postgreSQL.

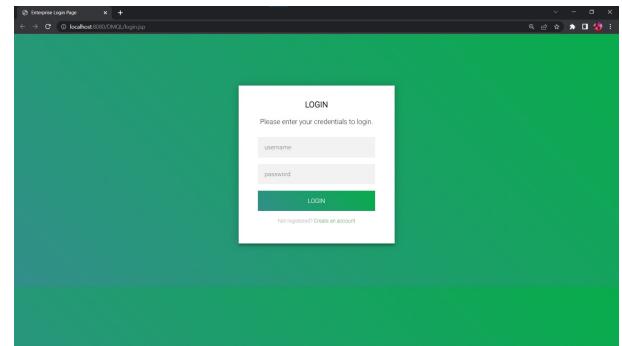
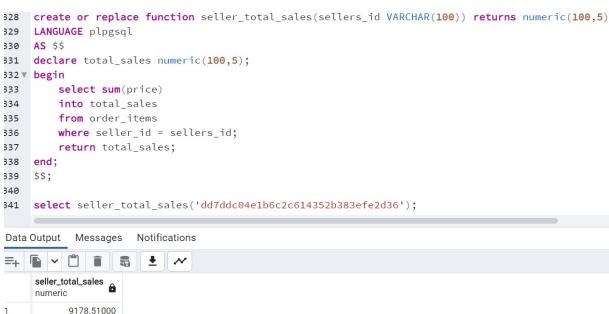


Fig. 46. Existing customer login



Fig. 47. New customer Registration



```

328 create or replace function seller_total_sales(sellers_id VARCHAR(100)) returns numeric(100,5)
329 LANGUAGE plpgsql
330 AS $$
331 declare total_sales numeric(100,5);
332 begin
333     select sum(price)
334     into total_sales
335     from order_items
336     where seller_id = sellers_id;
337     return total_sales;
338 end;
339 $$;
340
341 select seller_total_sales('dd7ddc04e1b6c2c614352b383eфе2d36');

```

Data Output Messages Notifications

| | seller_total_sales | numeric |
|---|--------------------|------------|
| 1 | | 9178.51000 |

Fig. 45. Function Implementation



Fig. 48. Actions that can be performed

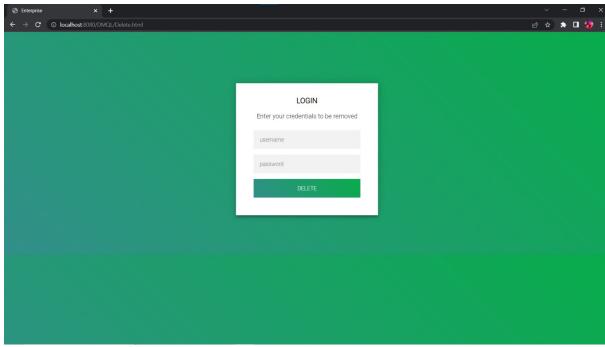


Fig. 49. Deleting user from the database



Fig. 50. Customers searching for the products to purchase

| Mobile Phone Details | | | |
|----------------------|--|--|-------------|
| product_id | product_name | description | price_image |
| 1 | Sony EX Ear Bud Headphones In White - MDREX2ULPWH | Sony EX Ear Bud Headphones In White - MDREX2ULPWH | 28.9 |
| 14 | Sony Universal Remote Commandes Remote Control - RMV310 | Sony Universal Remote Commandes Remote Control - RMV310 | 49.9 |
| 20 | Sony HD-HDMIcable 1.5 Meter (5 Feet) HDMI Max Cable - VMC15MBD | Sony HD-HDMIcable 1.5 Meter (5 Feet) HDMI Max Cable - VMC15MBD | 25.9 |
| 25 | Sony DVP-FX520 Red & Portable DVD Player - DVPFX520R | Sony DVP-FX520 Red & Portable DVD Player - DVPFX520R | 48.9 |
| 32 | Sony Son MixDVD-R Canscoode Media 3 Pack - JDMR30RH3 | Sony Son MixDVD-R Canscoode Media 3 Pack - JDMR30RH3 | 59.9 |
| 56 | Sony Universal Remote Control - RMEZ4 | Sony Universal Remote Control - RMEZ4 | 16.5 |
| 58 | Sony Silver 1080p Upscaling 5 Disc DVD Player - DVTPNC90HS | Sony Silver 1080p Upscaling 5 Disc DVD Player - DVTPNC90HS | 19.0 |
| 66 | Sony 27 BRAVIA V-Series Black LCD Flat Panel HDTV - KDE52N4100 | Sony 27 BRAVIA V-Series Black LCD Flat Panel HDTV - KDE52N4100 | 54.0 |
| 48 | Sony Digital Photo Printer Paper 120 Pack - SVMAF120P | Sony Digital Photo Printer Paper 120 Pack - SVMAF120P | 99.0 |
| 56 | Sony HD DVC Tape - DVSM6HD | Sony HD DVC Tape - DVSM6HD | 29.99 |
| 59 | Sony Soft Camera Carrying Case - LCSIMX100 | Sony Soft Camera Carrying Case - LCSIMX100 | 14.99 |
| 60 | Sony 16GB Memory Stick PRO Duo Mark 2 Media Card - MSMT16G | Sony 16GB Memory Stick PRO Duo Mark 2 Media Card - MSMT16G | 149.9 |

Fig. 51. Search images for the word 'Sony'

REFERENCES

- 1) <https://www.kaggle.com/datasets/olistbr/brazilian-e-commerce>
- 2) <https://www.w3schools.com/sql/>
- 3) <https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>
- 4) <https://extendsclass.com/csv-generator.html>
- 5) <https://www.eclipse.org/downloads/packages/release/neon/3/eclipse-ide-java-ee-developers>
- 6) <https://tomcat.apache.org/download-90.cgi>
- 7) <https://www.postgresqltutorial.com/postgresql-jdbc/connecting-to-postgresql-database/>