

Name: Prerana Shukla

Roll. No.: 31

Title: Inventory Store management System

Sem.: 5th Branch: ICT

Subject: Database Management System

Subject Code: 01CT0502

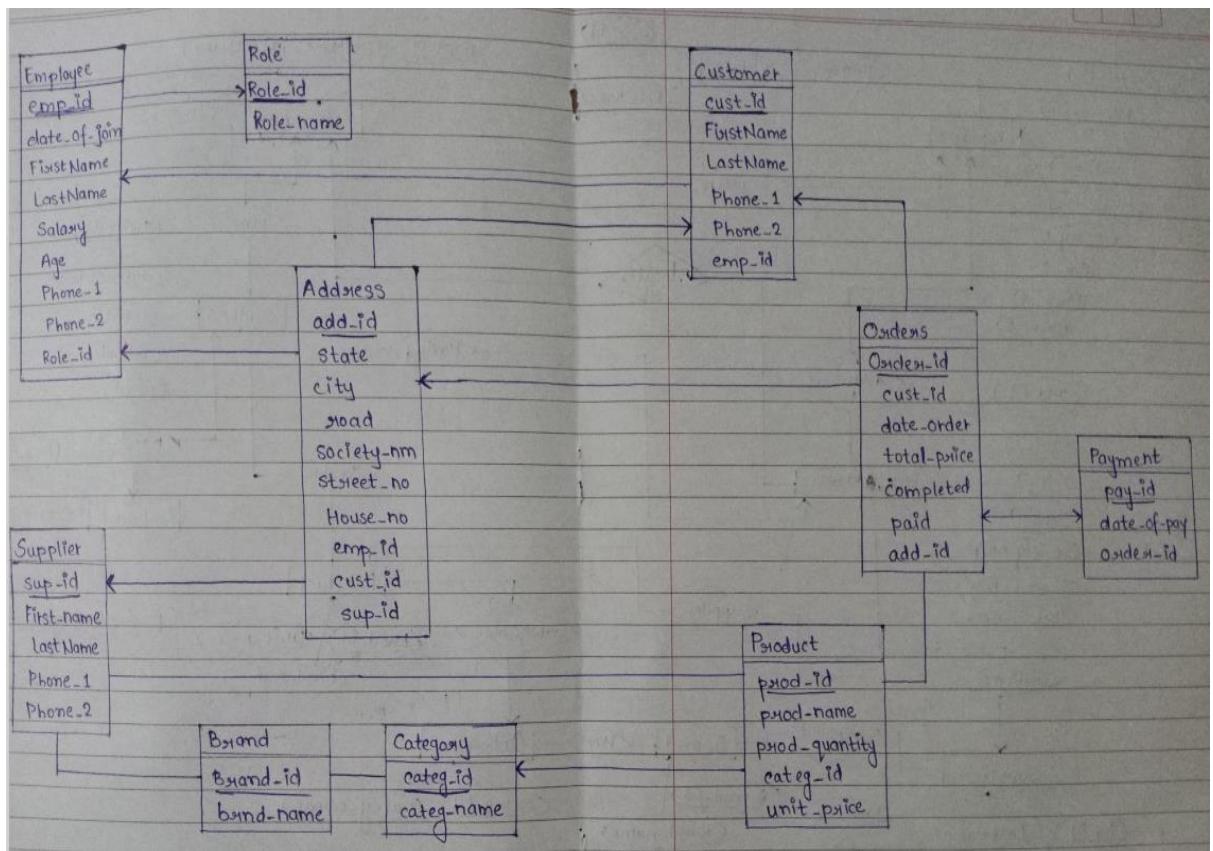
→ Abstract:

I have the project title as Inventory Store Management System. For the project I am going to build up a database creating table to store data created keeping in mind the need for inventory store to keep record and to retrieve it easily on time. I will use PostgreSQL pgAdmin4 for making and managing the database. Database Management System is a software that is designed to define, manipulate, retrieve and manage data in database. Database helps one to store all the logically related data and by using database management system it's easy for one for updating the database as and when needed, not only this it also gives flexibility to retrieve data if required. DBMS can be said as computerized record keeping system with the advancement it also provides a wide range to store data on cloud. I will use DBMS for the project Inventory Store Management System. While going through this I will follow the basic steps required for making the database in structured manner. Making schema diagram, implementing same with ER diagram for more clarity, schema designing which helps to have a glance for writing the query for er diagram and looking for cardinality constraints. Advantages of making a DBMS is that it reduces data redundancy so that duplication of a record can be neglected. As DBMS have atomicity so there is no ratio answer which provides better understanding of it, like if there is a customer asking for any product then it can be viewed that how much is the availability of that particular product there can be no half answer like. While not only this it allows applying integrity constraints, sharing of data and authorization of user such as the employee table can be seen by owner but not by customer.

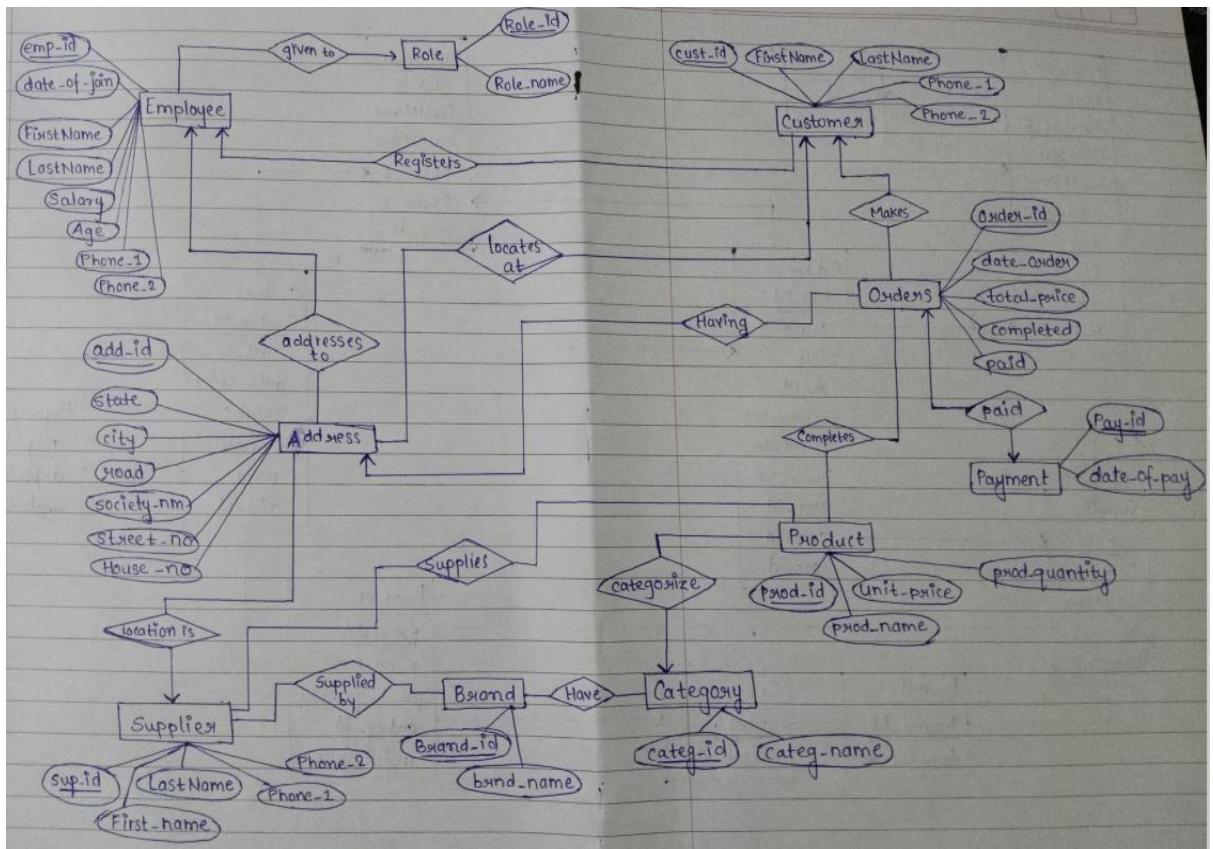
→ Schema Diagram:

Before going through the schema diagram lets define the basic concept for the same. Database schema is a skeleton structure which represents logical view of the entire database. Database schema defines entities and relationship among them which contains descriptive detail of the database that can depicted through schema diagram. Schema is designed by database designers.

Schema diagram is a diagram which contains entities and attributes that defines the schema. Need to notice that schema diagram only shows database design not the actual data of database.



→ E-R Diagram:



→ Schema Design:

Schema design.

Date : _____
Page : _____

→ For table Role
Role (Role-id, Role-name)

→ For table Category
Category (Categ-id, categ-name)

→ For table Brand
Brand (Brand-id, brand-name)

→ For table Employee
◦ as employee table have many to one cardinality constraints with table Role, so will make Role-id as foreign key in Employee Table

Employee (emp-id, date-of-join, FirstName,
LastName, Salary, Age, Phone-1, Phone-2,
Role-id)

→ For table Customer

◦ Here customer table have many to one cardinality constraint with Employee Table.
• So employee's Table primary key is set as Foreign Key in Customer Table.

Customer (cust-id, FirstName, LastName, Phone-1,
Phone-2, emp-id)

→ For Table Supplier

Supplier (sup-id, First-name, Last Name,
Phone-1, Phone-2)

→ For Table Product

- as Table product have many to one cardinality constraint with table category so categ-id (primary key of category Table) is set as Foreign key for Table Product.

Product (prod-id, prod-name, unit-price,
prod-quantity, categ-id)

→ For Table Address

- Many to one cardinality constraint with Employee, Customer, Supplier Table.

Address (add-id, state, city, road, society-nm,
street-no, house-no, emp-id, cust-id, sup-id)

→ For table Orders

Many to one cardinality constraints with customer and address Table.

Orders (order-id, date-order, total-price,
completed, paid, cust-id, add-id)

→ For table Paid Payment
as payment and orders table have one to one cardinality constraint, I have chosen to make order-id (primary key of Table Orders) to be foreign key for Table Payment, it can also be chosen vice versa.

Payment (pay-id , date-of-pay, order-id)

→ The Relationship between Supplier and Brand is many to many cardinality constraint, so need to make/take relationship as new Table make a unique id and take other two connected table's primary key as Foreign key for this new Table made.

Suppliedby (suppliedby-id , brand-id , sup-id)

→ Same way for other N:N Relationships

◦ Supplies (supplies-id , prod-id , sup-id ,
supplies-quantity , total-price)

◦ Have (Have-id , categ-id , brand-id)

◦ Completes (completes-id , prod-id , order-id ,
ordered-quantity)

→ Snapshots:

The screenshot shows the pgAdmin 4 interface with a connection to the database 'inventorystoremanagementstore_31'. The left sidebar displays various database objects like Schemas, Tables, and Views. The main area shows a query editor with the following SQL code:

```
1 set search_path to 'public';
2
3 CREATE TABLE Employee(emp_id varchar(20) NOT NULL UNIQUE, date_of_join_date, FirstName varchar(30), LastName varchar(30),
4                      Salary Float(2), Age Integer DEFAULT 25 CHECK(Age>23), Phone_1 BIGINT NOT NULL UNIQUE, Phone_2 BIGINT, Role_id varchar(20),
5                      PRIMARY KEY(emp_id), FOREIGN KEY(Role_id) REFERENCES Role(Role_id) );
6
7 CREATE TABLE Role(Role_id varchar(20) NOT NULL UNIQUE, Role_name varchar(20), PRIMARY KEY(Role_id));
8
9 CREATE TABLE Customer(cust_id varchar(20) NOT NULL UNIQUE, FirstName varchar(30), LastName varchar(30), Phone_1 BIGINT NOT NULL UNIQUE,
10                      Phone_2 BIGINT, emp_id varchar(20), PRIMARY KEY(cust_id), FOREIGN KEY(emp_id) REFERENCES Employee(emp_id) );
11
12 CREATE TABLE Orders(Order_id varchar(20) NOT NULL UNIQUE, cust_id varchar(20), add_id varchar(20), date_order date, total_price Integer DEFAULT 0,
13                      completed varchar(5) DEFAULT 'NO', paid varchar(5) DEFAULT 'NO', PRIMARY KEY(Order_id),
14                      FOREIGN KEY(cust_id) REFERENCES Customer(cust_id), FOREIGN KEY(add_id) REFERENCES Address(add_id));
15 --UPDATING THE TOTAL PRICE TO FLOAT
16 ALTER TABLE Orders ALTER COLUMN total_price TYPE Float(2);
17
18 CREATE TABLE Payment(pay_id varchar(20) NOT NULL UNIQUE, date_of_pay date, order_id varchar(20),
19                      PRIMARY KEY(pay_id), FOREIGN KEY(order_id) REFERENCES Orders(order_id));
20
21 CREATE TABLE Product(prod_id varchar(20) NOT NULL UNIQUE, prod_name varchar(20), unit_price Float(2), prod_quantity Integer,
22                      categ_id varchar(20), PRIMARY KEY(prod_id),
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Header:** 127.0.0.1:53442/browser/
- Toolbar:** File ▾ Object ▾ Tools ▾ Help ▾
- Sidebar:** Browser, Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Schemas (1), public, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Sequences, Tables (14), address, brand, category, completes, customer, employee, have, orders, payment, product, role.
- Query Editor:** inventorystoremanagementsystem_31/postgres@PostgreSQL 10
Query Editor | Query History
- Code:** The code is a series of CREATE TABLE statements defining the schema for an inventory management system. It includes tables for Product, Completes, Category, Brand, Supplies, and Supplier, along with their respective relationships and constraints.
- Bottom Navigation:** Data Output, Explain, Messages, Notifications
- Status Bar:** SET
Query returned successfully in 134 msec.

```
CREATE TABLE Suppliedby(suppliedby_id varchar(20) NOT NULL UNIQUE, Brand_id varchar(20), sup_id varchar(20),
PRIMARY KEY(suppliedby_id), FOREIGN KEY(Brand_id) REFERENCES Brand(Brand_id),
FOREIGN KEY(sup_id) REFERENCES Supplier(sup_id));

CREATE TABLE ADDRESS(add_id varchar(20) NOT NULL UNIQUE, state varchar(15), city varchar(15), road varchar(25), society_nm varchar(20),
street_no varchar(10), House_no varchar(10), emp_id varchar(20), cust_id varchar(20), sup_id varchar(20),
PRIMARY KEY(add_id), FOREIGN KEY(emp_id) REFERENCES Employee(emp_id), FOREIGN KEY(cust_id) REFERENCES Customer(cust_id),
FOREIGN KEY(sup_id) REFERENCES Supplier(sup_id));

--INSERTING VALUES TO ROLE:
INSERT INTO Role(Role_id, role_name) VALUES('R1', 'Manager');
INSERT INTO Role(Role_id, role_name) VALUES('R2', 'Common_Employee');
INSERT INTO Role(Role_id, role_name) VALUES('R3', 'Seller');
INSERT INTO Role(Role_id, role_name) VALUES('R4', 'Marketing_Executive');
INSERT INTO Role(Role_id, role_name) VALUES('R5', 'Guider');

--INSERTING VALUES IN CATEGORY
INSERT INTO Category(categ_id, categ_name) VALUES('C1', 'Dry Fruits');
INSERT INTO Category(categ_id, categ_name) VALUES('C2', 'Cold Drinks');
INSERT INTO Category(categ_id, categ_name) VALUES('C3', 'Food Packets');
INSERT INTO Category(categ_id, categ_name) VALUES('C4', 'Stationaries');
INSERT INTO Category(categ_id, categ_name) VALUES('C5', 'Milk Products');
```

pgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

```

64 INSERT INTO Category(categ_id, categ_name) VALUES('C5', 'Milk Products');
65 INSERT INTO Category(categ_id, categ_name) VALUES('C6', 'Pulses');
66 INSERT INTO Category(categ_id, categ_name) VALUES('C7', 'Grains');

67 --INSERTING VALUES IN EMPLOYEE
68 INSERT INTO Employee(emp_id, date_of_join, FirstName, LastName, Salary, Age, Phone_1, Phone_2, Role_id)
VALUES('E1', '2020-09-16', 'Aakash', 'Sharma', 25000.0, 26, 7826541253, 8526971594, 'R2' );
69 INSERT INTO Employee(emp_id, date_of_join, FirstName, LastName, Salary, Age, Phone_1, Role_id)
VALUES('E2', '2016-10-8', 'Prakash', 'Tripathi', 50000.0, 32, 7536913845, 'R1' );
70 INSERT INTO Employee(emp_id, date_of_join, FirstName, LastName, Salary, Phone_1, Role_id)
VALUES('E3', '2018-02-12', 'Mitul', 'Gajendra', 25000.0, 28, 7826518365,'R2' );
71 INSERT INTO Employee(emp_id, date_of_join, FirstName, LastName, Salary, Age, Phone_1, Phone_2, Role_id)
VALUES('E4', '2013-05-16', 'Neelam', 'Shah', 30000.0, 29, 8426917634, 8526971594, 'R3' );
72
73 --INSERTING VALUES IN CUSTOMER
74 INSERT INTO Customer(cust_id, FirstName, LastName, Phone_1, Phone_2, emp_id)
VALUES('C1', 'Aakash', 'Tripathi', 9825836751, 7852697432, 'E1' );
75 INSERT INTO Customer(cust_id, FirstName, LastName, Phone_1, emp_id)
VALUES('C2', 'Lalit', 'Kiplani', 8426873156,'E2' );
76 INSERT INTO Customer(cust_id, FirstName, LastName, Phone_1, Phone_2, emp_id)
VALUES('C3', 'Suman', 'Patel', 7526438512, 7852697432, 'E4' );
77
78 --Data Output Explain Messages Notifications
79 SET
80
81 Query returned successfully in 134 msec.

```

pgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

```

87 --INSERTING VALUES IN BRAND
88 INSERT INTO Brand(brnd_id, Brnd_name) VALUES('BR1', 'Balaji');
89 INSERT INTO Brand(brnd_id, Brnd_name) VALUES('BR2', 'Everest');
90 INSERT INTO Brand(brnd_id, Brnd_name) VALUES('BR3', 'Wilton');
91 INSERT INTO Brand(brnd_id, Brnd_name) VALUES('BR4', 'Classmate');
92 INSERT INTO Brand(brnd_id, Brnd_name) VALUES('BR5', 'Navneet');
93
94
95 --INSERTING VALUES IN HAVE
96 INSERT INTO Have(Have_id, categ_id, Brand_id) VALUES('H11', 'C1','BR1');
97 INSERT INTO Have(Have_id, categ_id, Brand_id) VALUES('H12', 'C4','BR3');
98 INSERT INTO Have(Have_id, categ_id, Brand_id) VALUES('H13', 'C5','BR2');
99 INSERT INTO Have(Have_id, categ_id, Brand_id) VALUES('H14', 'C2','BR4');
100 INSERT INTO Have(Have_id, categ_id, Brand_id) VALUES('H15', 'C6','BR5');
101
102
103 --INSERTING VALUES IN SUPPLIER
104 INSERT INTO Supplier(sup_id, First_name, LastName, Phone_1, Phone_2)
VALUES('SU1', 'Ravindar', 'Sharma', 7624584621, 8426137942);
105 INSERT INTO Supplier(sup_id, First_name, LastName, Phone_1)
VALUES('SU2', 'Divya', 'Chauhan', 9826475234);
106 INSERT INTO Supplier(sup_id, First_name, LastName, Phone_1, Phone_2)
VALUES('SU3', 'Kamlesh', 'Bhatt', 964251732, 8426137942);
107
108
109
110
111 --SELECT * FROM Supplier;
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
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
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1190
1191
1192
1193
1194
1195
1196
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1290
1291
1292
1293
1294
1295
1296
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1390
1391
1392
1393
1394
1395
1396
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1495
1496
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1595
1596
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1695
1696
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1790
1791
1792
1793
1794
1795
1796
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
21
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Left Panel (Browser Tree):** Shows the database structure with nodes like 'Inventorystoremanagementsyst' (selected), Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Schemas (1) (with public, collations, domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Sequences, and Tables (14) selected), address, brand, category, completes, customer, employee, have, orders, payment, product, and rma.
- Top Bar:** File, Object, Tools, Help.
- Toolbar:** Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, inventorystoremanagementstore_31.sql.
- Query Editor:** Contains the following SQL code:

```
129 --INSERTING INTO SUPPLIES
130 INSERT INTO Supplies(supplies_id, prod_id, sup_id, supplies_quantity, Total_price)
131     VALUES('SP1', 'PR1', 'SUI', 20, 50.0);
132
133 --UPDATE QUANTITY IN PRODUCT AS THE SUPPLY OF THAT PRODUCT IS ENTERED.
134 UPDATE Product SET prod_quantity = (SELECT ((SELECT prod_quantity FROM Product,Supplies WHERE Product.prod_id = Supplies.prod_id and Supplies.supplies_id = 'SP1')) FROM Supplies,Product WHERE supplies.prod_id = Product.prod_id and Supplies.supplies_id = 'SP1'));
135 WHERE prod_id = (SELECT Product.product_id FROM Product,Supplies WHERE Product.prod_id = Supplies.prod_id and Supplies.supplies_id = 'SP1');
136
137 INSERT INTO Supplies(supplies_id, prod_id, sup_id, supplies_quantity, Total_price)
138     VALUES('SP2', 'PR3', 'SUI', 2, 30.0);
139 --UPDATE QUANTITY IN PRODUCT AS THE SUPPLY OF THAT PRODUCT IS ENTERED.
140 UPDATE Product SET prod_quantity = (SELECT ((SELECT prod_quantity FROM Product,Supplies WHERE Product.prod_id = Supplies.prod_id and Supplies.supplies_id = 'SP2') + (SELECT supplies_quantity FROM Supplies,Product WHERE supplies.prod_id = Product.prod_id and Supplies.supplies_id = 'SP2'))) WHERE prod_id = (SELECT Product.product_id FROM Product,Supplies WHERE Product.prod_id = Supplies.prod_id and Supplies.supplies_id = 'SP2'));
141
142
143
144
145
146
147 INSERT INTO Supplies(supplies_id, prod_id, sup_id, supplies_quantity, Total_price)
148     VALUES('SP3', 'PR1', 'SUS', 30, 60.0);
149 --UPDATE QUANTITY IN PRODUCT AS THE SUPPLY OF THAT PRODUCT IS ENTERED.
150 UPDATE Product SET prod_quantity = (SELECT ((SELECT prod_quantity FROM Product,Supplies WHERE Product.prod_id = Supplies.prod_id and Supplies.supplies_id = 'SP3') + (SELECT supplies_quantity FROM Supplies,Product WHERE supplies.prod_id = Product.prod_id and Supplies.supplies_id = 'SP3'))) WHERE prod_id = (SELECT Product.product_id FROM Product,Supplies WHERE Product.prod_id = Supplies.prod_id and Supplies.supplies_id = 'SP3');
```

Bottom status bar: Data Output, Explain, Messages, Notifications. Message: Query returned successfully in 134 msec.

The screenshot shows the pgAdmin 4 interface with the following details:

- Left Sidebar:** Shows the database structure under "Inventorystoremanagementsyst".
 - Schemas:** Schemas (1) - public
 - Tables:** Tables (14) - address, brand, category, completes, customer, employee, have, orders, payment, product, rma
- Top Bar:** File, Object, Tools, Help
- Toolbar:** Browser, Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, inventorystoremanagementstore_31.sql
- Query Editor:** Contains the following SQL code:

```
147 INSERT INTO Supplies(supplies_id, prod_id, sup_id, supplies_quantity, Total_price)
148   VALUES('SP3', 'PR1', 'SU3', 30, 60.0);
149   --UPDATE QUANTITY IN PRODUCT AS THE SUPPLY OF THAT PRODUCT IS ENTERED.
150 UPDATE Product SET prod_quantity = (SELECT ((SELECT prod.quantity FROM Product,Supplies WHERE Product.prod_id = Supplies.prod_id and
151 Supplies.supplies_id = 'SP3')+(SELECT supplies_quantity FROM Supplies,Product WHERE supplies.prod_id =Product.prod_id
152 and Supplies.supplies_id = 'SP3'))) WHERE prod_id = (SELECT Product.prod_id FROM Product,Supplies
153 WHERE Product.prod_id = Supplies.prod_id and Supplies.supplies_id = 'SP3');
154
155
156 INSERT INTO Supplies(supplies_id, prod_id, sup_id, supplies_quantity, Total_price)
157   VALUES('SP4', 'PR2', 'SU2', 50, 20.0);
158   --UPDATE QUANTITY IN PRODUCT AS THE SUPPLY OF THAT PRODUCT IS ENTERED.
159 UPDATE Product SET prod_quantity = (SELECT ((SELECT prod.quantity FROM Product,Supplies WHERE Product.prod_id = Supplies.prod_id and
160 Supplies.supplies_id = 'SP4')+(SELECT supplies_quantity FROM Supplies,Product WHERE supplies.prod_id =Product.prod_id
161 and Supplies.supplies_id = 'SP4'))) WHERE prod_id = (SELECT Product.prod_id FROM Product,Supplies
162 WHERE Product.prod_id = Supplies.prod_id and Supplies.supplies_id = 'SP4');
163
164   --SEEING UPDATED PRODUCT TABLE AND PRODUCT QUANTITY
165 SELECT * FROM Product;
166
167   --INSERTING INTO ADDRESS
168 INSERT INTO Address(add_id, state, city, road, society_nm, street_no, House_no, emp_id)
```
- Data Output:** Shows the message "Query returned successfully in 134 msec."

The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** File, Object, Tools, Help.
- Left Sidebar:** Browser (expanded), Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Schemas (1), public (expanded), Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Sequences (1), Tables (14) (selected).
- Query Editor:** Query History tab selected. The code is as follows:

```
167 --INSERTING INTO ADDRESS
168 INSERT INTO Address(add_id, state, city, road, society_nm, street_no, House_no, emp_id)
169     VALUES('AD1', 'Gujarat', 'Rajkot', 'Jammagar Road', 'Global', 'V3', 'HNI', 'E1' );
170 INSERT INTO Address(add_id, state, city, road, society_nm, street_no, House_no, emp_id)
171     VALUES('AD2', 'Gujarat', 'Rajkot', 'Madhapar', 'Sun Rise', 'S2', 'SR1', 'E2' );
172 INSERT INTO Address(add_id, state, city, road, society_nm, street_no, House_no, emp_id)
173     VALUES('AD3', 'MP', 'Ujjain', 'Pragya', 'Prakash', 'P1', 'PR5', 'E3' );
174 INSERT INTO Address(add_id, state, city, road, society_nm, street_no, House_no, emp_id)
175     VALUES('AD4', 'Gujarat', 'Rajkot', 'Indira Circle', 'Crystal', 'C5', 'CR2', 'E4' );
176 INSERT INTO Address(add_id, state, city, road, society_nm, street_no, House_no, emp_id)
177     VALUES('AD5', 'Gujarat', 'Rajkot', 'Raiya Road', 'Silver Height', 'SH4', 'H1', 'E4' );
178
179 INSERT INTO Address(add_id, state, city, road, society_nm, street_no, House_no, cust_id)
180     VALUES('AD6', 'Gujarat', 'Rajkot', 'Jammagar Road', 'COPPER CITY', 'CC5', 'HN60', 'CU1' );
181 INSERT INTO Address(add_id, state, city, road, society_nm, street_no, House_no, cust_id)
182     VALUES('AD7', 'MP', 'Lucknow', 'Sabar Road', 'HighLights', 'HG2', 'HN120', 'CU2' );
183 INSERT INTO Address(add_id, state, city, road, society_nm, street_no, House_no, cust_id)
184     VALUES('AD8', 'Gujarat', 'Surat', 'Surat Highway', 'Sun Raise', 'S5', 'SRE30', 'CU3' );
185
186 INSERT INTO Address(add_id, state, city, road, society_nm, street_no, House_no, sup_id)
187     VALUES('AD9', 'Gujarat', 'Rajkot', 'Raiya Exchange', 'Ramananda', 'R1', 'R10', 'SU1' );
188 INSERT INTO Address(add_id, state, city, road, society_nm, street_no, House_no, sup_id)
```

Bottom status bar: Data Output, Explain, Messages, Notifications.

PGAdmin - inventorystoremanagementsyst

```

186 INSERT INTO Address(add_id, state, city, road, society_nm, street_no, House_no, sup_id)
187     VALUES('AD9', 'Gujarat', 'Rajkot', 'Raiya Exchange', 'Ramananda', 'R1', 'R10', 'SU1')
188 INSERT INTO Address(add_id, state, city, road, society_nm, street_no, House_no, sup_id)
189     VALUES('AD10', 'MP', 'Ujjain', 'Gopalpura Road', 'Nivti', 'N2', 'NV28', 'SU2')
190 INSERT INTO Address(add_id, state, city, road, society_nm, street_no, House_no, sup_id)
191     VALUES('AD11', 'Gujarat', 'Rajkot', 'Hospital Chowk', 'High Pillars', 'H3', 'HP32', 'SU3')
192
193 --INSERTING VALUES IN ORDERS
194 INSERT INTO Orders(order_id, cust_id, add_id, date_order) VALUES('OD1', 'CU1', 'AD6', '2019-09-21');
195 INSERT INTO Orders(order_id, cust_id, add_id, date_order) VALUES('OD2', 'CU2', 'AD7', '2020-10-24');
196 INSERT INTO Orders(order_id, cust_id, add_id, date_order) VALUES('OD3', 'CU3', 'AD8', '2018-1-5');
197 INSERT INTO Orders(order_id, cust_id, add_id, date_order) VALUES('OD4', 'CU4', 'AD7', '2021-10-25');
198 DELETE FROM Orders WHERE date_order = '2021-10-25';
199 INSERT INTO Orders(order_id, cust_id, add_id, date_order) VALUES('OD4', 'CU2', 'AD7', '2021-1-25');
200
201 INSERT INTO Orders(order_id, cust_id, add_id, date_order) VALUES('OD5', 'CU2', 'AD7', '2021-5-22');
202
203 --INSERTING VALUES IN COMPLETES
204 INSERT INTO Completes(completes_id, prod_id, order_id, ordered_quantity)
205     VALUES('CMP1', 'PR1', 'OD1', 10);
206 --UPDATE QUANTITY IN PRODUCT AS THE SUPPLY OF THAT PRODUCT IS ENTERED.
207 UPDATE Product SET prod.quantity = (SELECT ((SELECT prod.quantity FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND
208     Completes.completes_id = 'CMP1') - (SELECT ordered_quantity FROM Completes,Product WHERE completes.prod_id = Product.prod_id
209         AND Completes.completes_id = 'CMP1'))) WHERE prod_id = (SELECT Product.prod_id FROM Product,Completes
210         WHERE Product.prod_id = completes.prod_id AND Completes.completes_id = 'CMP1');
211
212 --UPDATING THAT ORDER COMPLETED
213 UPDATE Orders SET completed = 'YES' WHERE order_id = (SELECT order_id FROM Completes WHERE completes_id = 'CMP1');
214
215 --UPDATING FOR THE PRICE
216 UPDATE Orders SET total_price = (SELECT((SELECT total_price FROM Orders,Completes WHERE Orders.order_id = Completes.order_id AND
217     Completes.completes_id = 'CMP1') + (SELECT (SELECT unit_price FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND
218         Completes.completes_id = 'CMP1') * (SELECT ordered_quantity FROM Completes WHERE completes_id = 'CMP1'))));
219 WHERE order_id =
220     (SELECT Orders.order_id FROM Orders, Completes WHERE Orders.order_id = Completes.order_id AND Completes.completes_id = 'CMP1');
221
222 INSERT INTO Completes(completes_id, prod_id, order_id, ordered_quantity)
223     VALUES('CMP2', 'PR3', 'OD1', 2);
224
225 --UPDATE QUANTITY IN PRODUCT AS THE SUPPLY OF THAT PRODUCT IS ENTERED.
226 UPDATE Product SET prod.quantity = (SELECT ((SELECT prod.quantity FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND
227     Completes.completes_id = 'CMP2') - (SELECT ordered_quantity FROM Completes,Product WHERE completes.prod_id = Product.prod_id
228         AND Completes.completes_id = 'CMP2'))) WHERE prod_id = (SELECT Product.prod_id FROM Product,Completes
229         WHERE Product.prod_id = completes.prod_id AND Completes.completes_id = 'CMP2');
230
231 --UPDATING THAT ORDER COMPLETED
232 UPDATE Orders SET completed = 'YES' WHERE order_id = (SELECT order_id FROM Completes WHERE completes_id = 'CMP2');
233 --UPDATING FOR THE PRICE
234 UPDATE Orders SET total_price = (SELECT((SELECT total_price FROM Orders,Completes WHERE Orders.order_id = Completes.order_id AND
235     Completes.completes_id = 'CMP2') + (SELECT (SELECT unit_price FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND
236         Completes.completes_id = 'CMP2') * (SELECT ordered_quantity FROM Completes WHERE completes_id = 'CMP2'))));
237 WHERE order_id =
238     (SELECT Orders.order_id FROM Orders, Completes WHERE Orders.order_id = Completes.order_id AND Completes.completes_id = 'CMP2');
239
240 INSERT INTO Completes(completes_id, prod_id, order_id, ordered_quantity)
241     VALUES('CMP3', 'PR1', 'OD2', 22);
242
243 --UPDATE QUANTITY IN PRODUCT AS THE SUPPLY OF THAT PRODUCT IS ENTERED.
244 UPDATE Product SET prod.quantity = (SELECT ((SELECT prod.quantity FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND
245     Completes.completes_id = 'CMP3') - (SELECT ordered_quantity FROM Completes,Product WHERE completes.prod_id = Product.prod_id
246         AND Completes.completes_id = 'CMP3')) WHERE prod_id = (SELECT Product.prod_id FROM Product,Completes
247         WHERE Product.prod_id = completes.prod_id AND Completes.completes_id = 'CMP3'));

```

Data Output Explain Messages Notifications

SET

Query returned successfully in 134 msec.

PGAdmin - inventorystoremanagementsyst

```

203 --INSERTING VALUES IN COMPLETES
204 INSERT INTO Completes(completes_id, prod_id, order_id, ordered_quantity)
205     VALUES('CMP1', 'PR1', 'OD1', 10);
206 --UPDATE QUANTITY IN PRODUCT AS THE SUPPLY OF THAT PRODUCT IS ENTERED.
207 UPDATE Product SET prod.quantity = (SELECT ((SELECT prod.quantity FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND
208     Completes.completes_id = 'CMP1') - (SELECT ordered_quantity FROM Completes,Product WHERE completes.prod_id = Product.prod_id
209         AND Completes.completes_id = 'CMP1'))) WHERE prod_id = (SELECT Product.prod_id FROM Product,Completes
210         WHERE Product.prod_id = completes.prod_id AND Completes.completes_id = 'CMP1');
211
212 --UPDATING THAT ORDER COMPLETED
213 UPDATE Orders SET completed = 'YES' WHERE order_id = (SELECT order_id FROM Completes WHERE completes_id = 'CMP1');
214
215 --UPDATING FOR THE PRICE
216 UPDATE Orders SET total_price = (SELECT((SELECT total_price FROM Orders,Completes WHERE Orders.order_id = Completes.order_id AND
217     Completes.completes_id = 'CMP1') + (SELECT (SELECT unit_price FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND
218         Completes.completes_id = 'CMP1') * (SELECT ordered_quantity FROM Completes WHERE completes_id = 'CMP1'))));
219 WHERE order_id =
220     (SELECT Orders.order_id FROM Orders, Completes WHERE Orders.order_id = Completes.order_id AND Completes.completes_id = 'CMP1');
221
222 INSERT INTO Completes(completes_id, prod_id, order_id, ordered_quantity)
223     VALUES('CMP2', 'PR3', 'OD1', 2);
224
225 --UPDATE QUANTITY IN PRODUCT AS THE SUPPLY OF THAT PRODUCT IS ENTERED.
226 UPDATE Product SET prod.quantity = (SELECT ((SELECT prod.quantity FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND
227     Completes.completes_id = 'CMP2') - (SELECT ordered_quantity FROM Completes,Product WHERE completes.prod_id = Product.prod_id
228         AND Completes.completes_id = 'CMP2'))) WHERE prod_id = (SELECT Product.prod_id FROM Product,Completes
229         WHERE Product.prod_id = completes.prod_id AND Completes.completes_id = 'CMP2');
230
231 --UPDATING THAT ORDER COMPLETED
232 UPDATE Orders SET completed = 'YES' WHERE order_id = (SELECT order_id FROM Completes WHERE completes_id = 'CMP2');
233 --UPDATING FOR THE PRICE
234 UPDATE Orders SET total_price = (SELECT((SELECT total_price FROM Orders,Completes WHERE Orders.order_id = Completes.order_id AND
235     Completes.completes_id = 'CMP2') + (SELECT (SELECT unit_price FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND
236         Completes.completes_id = 'CMP2') * (SELECT ordered_quantity FROM Completes WHERE completes_id = 'CMP2'))));
237 WHERE order_id =
238     (SELECT Orders.order_id FROM Orders, Completes WHERE Orders.order_id = Completes.order_id AND Completes.completes_id = 'CMP2');
239
240 INSERT INTO Completes(completes_id, prod_id, order_id, ordered_quantity)
241     VALUES('CMP3', 'PR1', 'OD2', 22);
242
243 --UPDATE QUANTITY IN PRODUCT AS THE SUPPLY OF THAT PRODUCT IS ENTERED.
244 UPDATE Product SET prod.quantity = (SELECT ((SELECT prod.quantity FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND
245     Completes.completes_id = 'CMP3') - (SELECT ordered_quantity FROM Completes,Product WHERE completes.prod_id = Product.prod_id
246         AND Completes.completes_id = 'CMP3')) WHERE prod_id = (SELECT Product.prod_id FROM Product,Completes
247         WHERE Product.prod_id = completes.prod_id AND Completes.completes_id = 'CMP3'));

```

Data Output Explain Messages Notifications

SET

Query returned successfully in 134 msec.

PGAdmin - inventorystoremanagementsyst

```

222 INSERT INTO Completes(completes_id, prod_id, order_id, ordered_quantity)
223     VALUES('CMP2', 'PR3', 'OD1', 2);
224
225 --UPDATE QUANTITY IN PRODUCT AS THE SUPPLY OF THAT PRODUCT IS ENTERED.
226 UPDATE Product SET prod.quantity = (SELECT ((SELECT prod.quantity FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND
227     Completes.completes_id = 'CMP2') - (SELECT ordered_quantity FROM Completes,Product WHERE completes.prod_id = Product.prod_id
228         AND Completes.completes_id = 'CMP2'))) WHERE prod_id = (SELECT Product.prod_id FROM Product,Completes
229         WHERE Product.prod_id = completes.prod_id AND Completes.completes_id = 'CMP2');
230
231 --UPDATING THAT ORDER COMPLETED
232 UPDATE Orders SET completed = 'YES' WHERE order_id = (SELECT order_id FROM Completes WHERE completes_id = 'CMP2');
233 --UPDATING FOR THE PRICE
234 UPDATE Orders SET total_price = (SELECT((SELECT total_price FROM Orders,Completes WHERE Orders.order_id = Completes.order_id AND
235     Completes.completes_id = 'CMP2') + (SELECT (SELECT unit_price FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND
236         Completes.completes_id = 'CMP2') * (SELECT ordered_quantity FROM Completes WHERE completes_id = 'CMP2'))));
237 WHERE order_id =
238     (SELECT Orders.order_id FROM Orders, Completes WHERE Orders.order_id = Completes.order_id AND Completes.completes_id = 'CMP2');
239
240 INSERT INTO Completes(completes_id, prod_id, order_id, ordered_quantity)
241     VALUES('CMP3', 'PR1', 'OD2', 22);
242
243 --UPDATE QUANTITY IN PRODUCT AS THE SUPPLY OF THAT PRODUCT IS ENTERED.
244 UPDATE Product SET prod.quantity = (SELECT ((SELECT prod.quantity FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND
245     Completes.completes_id = 'CMP3') - (SELECT ordered_quantity FROM Completes,Product WHERE completes.prod_id = Product.prod_id
246         AND Completes.completes_id = 'CMP3')) WHERE prod_id = (SELECT Product.prod_id FROM Product,Completes
247         WHERE Product.prod_id = completes.prod_id AND Completes.completes_id = 'CMP3'));

```

Data Output Explain Messages Notifications

SET

Query returned successfully in 134 msec.

127.0.0.1:5432/browser/

pgAdmin 4 - inventorystoremanagementsyst

```

Browser    File  Object  Tools  Help
Dashboard Properties SQL Statistics Dependencies Dependents inventorystoremanagementstore_31.sql
Query Editor Query History
243 --UPDATE QUANTITY IN PRODUCT AS THE SUPPLY OF THAT PRODUCT IS ENTERED.
244 UPDATE Product SET prod_quantity = (SELECT ((SELECT prod_quantity FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND Completes.completes_id = 'CMP3')) - (SELECT ordered_quantity FROM Completes,Product WHERE completes.prod_id = Product.prod_id AND Completes.completes_id = 'CMP3')) WHERE prod_id = (SELECT Product.prod_id FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND Completes.completes_id = 'CMP3');
245
246 --UPDATING THAT ORDER COMPLETED
247 UPDATE Orders SET completed = 'YES' WHERE order_id = (SELECT order_id FROM Completes WHERE completes_id = 'CMP3');
248 --UPDATING FOR THE PRICE
249 UPDATE Orders SET total_price = (SELECT((SELECT total_price FROM Orders,Completes WHERE Orders.order_id = Completes.order_id AND Completes.completes_id = 'CMP3')) + (SELECT (SELECT unit_price FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND Completes.completes_id = 'CMP3') * (SELECT ordered_quantity FROM Completes WHERE completes_id = 'CMP3')))) WHERE order_id =
250 (SELECT Orders.order_id FROM Orders, Completes WHERE Orders.order_id = Completes.order_id AND Completes.completes_id = 'CMP3');
251
252 --INSERT INTO Completes(completes_id, prod_id, order_id, ordered_quantity)
253 VALUES('CMP4', 'PR2','003',30);
254
255 --UPDATE QUANTITY IN PRODUCT AS THE SUPPLY OF THAT PRODUCT IS ENTERED.
256 UPDATE Product SET prod_quantity = (SELECT ((SELECT prod_quantity FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND Completes.completes_id = 'CMP4')) - (SELECT ordered_quantity FROM Completes,Product WHERE completes.prod_id = Product.prod_id AND Completes.completes_id = 'CMP4')) WHERE prod_id = (SELECT Product.prod_id FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND Completes.completes_id = 'CMP4');
257
258 --UPDATING THAT ORDER COMPLETED
259 Data Output Explain Messages Notifications
260 SET
261
262 Query returned successfully in 134 msec.

```

pgAdmin 4 - inventorystoremanagementsyst

```

Browser    File  Object  Tools  Help
Dashboard Properties SQL Statistics Dependencies Dependents inventorystoremanagementstore_31.sql
Query Editor Query History
266 --UPDATING THAT ORDER COMPLETED
267 UPDATE Orders SET completed = 'YES' WHERE order_id = (SELECT order_id FROM Completes WHERE completes_id = 'CMP4');
268 --UPDATING FOR THE PRICE
269 UPDATE Orders SET total_price = (SELECT((SELECT total_price FROM Orders,Completes WHERE Orders.order_id = Completes.order_id AND Completes.completes_id = 'CMP4')) + (SELECT (SELECT unit_price FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND Completes.completes_id = 'CMP4') * (SELECT ordered_quantity FROM Completes WHERE completes_id = 'CMP1')))) WHERE order_id =
270 (SELECT Orders.order_id FROM Orders, Completes WHERE Orders.order_id = Completes.order_id AND Completes.completes_id = 'CMP4');
271
272 --INSERT INTO Completes(completes_id, prod_id, order_id, ordered_quantity)
273 VALUES('CMP5', 'PR2','003',10);
274
275 --deleting value as placed wrong
276 DELETE FROM Completes WHERE order_id = '003';
277
278 INSERT INTO Completes(completes_id, prod_id, order_id, ordered_quantity)
279 VALUES('CMP5', 'PR2','004',10);
280
281 --UPDATE QUANTITY IN PRODUCT AS THE SUPPLY OF THAT PRODUCT IS ENTERED.
282 UPDATE Product SET prod_quantity = (SELECT ((SELECT prod_quantity FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND Completes.completes_id = 'CMP5')) - (SELECT ordered_quantity FROM Completes,Product WHERE completes.prod_id = Product.prod_id AND Completes.completes_id = 'CMP5')) WHERE prod_id = (SELECT Product.prod_id FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND Completes.completes_id = 'CMP5'));
283
284 --UPDATING THAT ORDER COMPLETED
285 Data Output Explain Messages Notifications
286 SET
287
288 Query returned successfully in 134 msec.

```

pgAdmin 4 - inventorystoremanagementsyst

```

Browser    File  Object  Tools  Help
Dashboard Properties SQL Statistics Dependencies Dependents inventorystoremanagementstore_31.sql
Query Editor Query History
289 --UPDATING THAT ORDER COMPLETED
290 UPDATE Orders SET completed = 'YES' WHERE order_id = (SELECT order_id FROM Completes WHERE completes_id = 'CMP5');
291 --UPDATING FOR THE PRICE
292 UPDATE Orders SET total_price = (SELECT((SELECT total_price FROM Orders,Completes WHERE Orders.order_id = Completes.order_id AND Completes.completes_id = 'CMP5')) + (SELECT (SELECT unit_price FROM Product,Completes WHERE Product.prod_id = Completes.prod_id AND Completes.completes_id = 'CMP5') * (SELECT ordered_quantity FROM Completes WHERE completes_id = 'CMP5')))) WHERE order_id =
293 (SELECT Orders.order_id FROM Orders, Completes WHERE Orders.order_id = Completes.order_id AND Completes.completes_id = 'CMP5');
294
295 --SEEING UPDATED ORDERS TABLE FOR SEEING WHICH ORDERS ARE COMPLETED
296 SELECT * FROM Orders;
297
298 --SEEING THE UPDATED QUANTITY FOR PRODUCT TABLE
299 SELECT * FROM Product;
300
301 --INSERTING VALUES IN PAYMENT
302 INSERT INTO Payment(pay_id, date_of_pay, order_id) VALUES('PAY1','2020-12-11','002');
303
304 --UPDATING ORDERS TABLE THAT PAYMENT CAME FOR THE ORDER
305 UPDATE Orders SET paid = 'YES' WHERE order_id = (SELECT order_id FROM Payment WHERE pay_id = 'PAY1');
306
307 --UPDATING ORDERS TABLE THAT PAYMENT CAME FOR THE ORDER
308 INSERT INTO Payment(pay_id, date_of_pay, order_id) VALUES('PAY2','2019-06-1','003');
309
310 --UPDATING ORDERS TABLE THAT PAYMENT CAME FOR THE ORDER
311
312 Data Output Explain Messages Notifications
313 SET
314
315 Query returned successfully in 134 msec.

```

PgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

- > Casts
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Schemas (1)
 - > public
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > Aa FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Tables (14)
 - > address
 - > brand
 - > category
 - > completes
 - > customer
 - > employee
 - > have
 - > orders
 - > payment
 - > product
 - > role

Query Editor inventorystoremanagementsystem_31/postgres@PostgreSQL 10

```

309
310 INSERT INTO Payment(payment_id, date_of_pay, order_id) VALUES('PAY2','2019-06-1','003');
311
312 --UPDATING ORDERS TABLE THAT PAYMENT CAME FOR THE ORDER
313 UPDATE Orders SET paid = 'YES' WHERE order_id = (SELECT order_id FROM Payment WHERE pay_id = 'PAY2');
314
315 INSERT INTO Payment(payment_id, date_of_pay, order_id) VALUES('PAY3','2020-04-15','001');
316
317 --UPDATING ORDERS TABLE THAT PAYMENT CAME FOR THE ORDER
318 UPDATE Orders SET paid = 'YES' WHERE order_id = (SELECT order_id FROM Payment WHERE pay_id = 'PAY3');
319
320 INSERT INTO Payment(payment_id, date_of_pay, order_id) VALUES('PAY4','2021-02-15','004');
321
322 --UPDATING ORDERS TABLE THAT PAYMENT CAME FOR THE ORDER
323 UPDATE Orders SET paid = 'YES' WHERE order_id = (SELECT order_id FROM Payment WHERE pay_id = 'PAY4');
324
325 --SEEING WHICH OF THE ORDERS ARE PAID
326 SELECT * FROM Orders;
327
328
329 --ALL TABLES
330 SELECT * FROM Employee;
331 SELECT * FROM Role;
332 SELECT * FROM Customer;
333
334
335
336
337

```

Data Output Explain Messages Notifications

SET

Query returned successfully in 134 msec.

TABLES:

PgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

- > Casts
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Schemas (1)
 - > public
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > Aa FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Tables (14)
 - > address
 - > brand
 - > category
 - > completes
 - > customer
 - > employee
 - > have
 - > orders

Query Editor inventorystoremanagementsystem_31/postgres@PostgreSQL 10

```

320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337

```

emp_id	date_of_join	firstname	lastname	salary	age	phone_1	phone_2	role_id
1 E1	2020-09-16	Ayush	Sharma	25000	26	7826541253	8526971594	R2
2 E2	2016-10-08	Praakash	Trivedi	50000	32	7536913845	[null]	R1
3 E3	2018-02-12	Mitul	Gajendra	25000	25	7826518365	[null]	R2
4 E4	2013-05-16	Neelam	Shah	30000	29	8426917634	8526971594	R3

PgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

- > Casts
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Schemas (1)
 - > public
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > Aa FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Tables (14)
 - > address
 - > brand
 - > category
 - > completes
 - > customer
 - > employee
 - > have
 - > orders
 - > payment

Query Editor inventorystoremanagementsystem_31/postgres@PostgreSQL 10

```

320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337

```

role_id	role_name
1 R1	Manager
2 R2	Common_Employee
3 R3	Seller
4 R4	Marketing Executive
5 R5	Guider

pgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

```
--> Casts
--> Catalogs
--> Event Triggers
--> Extensions
--> Foreign Data Wrappers
--> Languages
--> Schemas (1)
--> public
--> Collations
--> Domains
--> FTS Configurations
--> FTS Dictionaries
--> Aa FTS Parsers
--> FTS Templates
--> Foreign Tables
--> Functions
--> Materialized Views
--> Sequences
--> Tables (14)
--> address
--> brand
--> category
--> completes
--> customer
--> employee
```

Query Editor Query History

```
323 <DATE> ORDER WHERE ORDER_PAID = YES WHERE ORDER_ID = (SELECT ORDER_ID FROM PAYMENTS WHERE PAY_ID = 1014);
```

```
324
325 --SEEING WHICH OF THE ORDERS ARE PAID
326 SELECT * FROM Orders;
```

```
327
328
329 --ALL TABLES
330 SELECT * FROM Employee;
331 SELECT * FROM Role;
332 SELECT * FROM Customer;
333 SELECT * FROM Orders;
334 SELECT * FROM Payment;
335 SELECT * FROM Product;
336 SELECT * FROM Address;
337 SELECT * FROM Brand;
```

Data Output Explain Messages Notifications

cust_id	firstname	lastname	phone_1	phone_2	emp_id
CU1	Aakash	Tripathi	9825836751	7852697432	E1
CU2	Lalit	Kripalani	8426873156	[null]	E2
CU3	Suman	Patel	7526438512	7852697432	E4

pgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

```
--> Casts
--> Catalogs
--> Event Triggers
--> Extensions
--> Foreign Data Wrappers
--> Languages
--> Schemas (1)
--> public
--> Collations
--> Domains
--> FTS Configurations
--> FTS Dictionaries
--> Aa FTS Parsers
--> FTS Templates
--> Foreign Tables
--> Functions
--> Materialized Views
--> Sequences
--> Tables (14)
--> address
--> brand
--> category
--> completes
--> customer
--> employee
--> have
--> orders
--> payment
--> product
```

Query Editor Query History

```
323 <DATE> ORDER WHERE ORDER_PAID = YES WHERE ORDER_ID = (SELECT ORDER_ID FROM PAYMENTS WHERE PAY_ID = 1014);
```

```
324
325 --SEEING WHICH OF THE ORDERS ARE PAID
326 SELECT * FROM Orders;
```

```
327
328
329 --ALL TABLES
330 SELECT * FROM Employee;
331 SELECT * FROM Role;
332 SELECT * FROM Customer;
333 SELECT * FROM Orders;
334 SELECT * FROM Payment;
335 SELECT * FROM Product;
336 SELECT * FROM Address;
337 SELECT * FROM Brand;
```

Data Output Explain Messages Notifications

order_id	cust_id	add_id	date_order	total_price	completed	paid
OD5	CU2	AD7	2021-05-22	0	NO	
OD2	CU2	AD7	2020-10-24	440	YES	YES
OD3	CU3	AD8	2018-01-05	50	YES	YES
OD1	CU1	AD6	2019-09-21	220	YES	YES
OD4	CU2	AD7	2021-01-25	50	YES	YES

pgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

```
--> Casts
--> Catalogs
--> Event Triggers
--> Extensions
--> Foreign Data Wrappers
--> Languages
--> Schemas (1)
--> public
--> Collations
--> Domains
--> FTS Configurations
--> FTS Dictionaries
--> Aa FTS Parsers
--> FTS Templates
--> Foreign Tables
--> Functions
--> Materialized Views
--> Sequences
--> Tables (14)
--> address
--> brand
--> category
--> completes
--> customer
--> employee
--> have
```

Query Editor Query History

```
323 <DATE> ORDER WHERE ORDER_PAID = YES WHERE ORDER_ID = (SELECT ORDER_ID FROM PAYMENTS WHERE PAY_ID = 1014);
```

```
324
325 --SEEING WHICH OF THE ORDERS ARE PAID
326 SELECT * FROM Orders;
```

```
327
328
329 --ALL TABLES
330 SELECT * FROM Employee;
331 SELECT * FROM Role;
332 SELECT * FROM Customer;
333 SELECT * FROM Orders;
334 SELECT * FROM Payment;
335 SELECT * FROM Product;
336 SELECT * FROM Address;
337 SELECT * FROM Brand;
```

Data Output Explain Messages Notifications

pay_id	date_of_pay	order_id
PAY1	2020-12-11	OD2
PAY2	2019-06-01	OD3
PAY3	2020-04-15	OD1
PAY4	2021-02-15	OD4

pgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

Query Editor Query History

```

322 ORDER BY pay_id DESC LIMIT 1000 WHERE pay_id = (SELECT pay_id FROM payment WHERE pay_id = 1000);
324 --SEEING WHICH OF THE ORDERS ARE PAID
326 SELECT * FROM Orders;
327
328
329 --ALL TABLES
330 SELECT * FROM Employee;
331 SELECT * FROM Role;
332 SELECT * FROM Customer;
333 SELECT * FROM Orders;
334 SELECT * FROM Payment;
335 SELECT * FROM Product;
336 SELECT * FROM Address;
337 SELECT * FROM Brand;

```

Data Output Explain Messages Notifications

pay_id	[PK] character varying (20)	date_of_pay	order_id
1	PAY1	2020-12-11	OD2
2	PAY2	2019-06-01	OD3
3	PAY3	2020-04-15	OD1
4	PAY4	2021-02-15	OD4

pgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

Query Editor Query History

```

330 SELECT * FROM Employee;
331 SELECT * FROM Role;
332 SELECT * FROM Customer;
333 SELECT * FROM Orders;
334 SELECT * FROM Payment;
335 SELECT * FROM Product;
336 SELECT * FROM Address;
337 SELECT * FROM Brand;
338 SELECT * FROM Category;
339 SELECT * FROM Completes;
340 SELECT * FROM Have;

```

Data Output Explain Messages Notifications

add_id	[PK] character varying (20)	state	city	road	society_nm	street_no	house_no
1	AD1	Gujarat	Rajkot	Jamnagar Road	Global	V3	HN1
2	AD2	Gujarat	Rajkot	Madhapor	Sun Rise	S2	SR1
3	AD3	MP	Ujjain	Pragya	Prakash	P1	PR5
4	AD4	Gujarat	Rajkot	Indira Circle	Crystal	C5	CR2
5	AD5	Gujarat	Rajkot	Raiya Road	Silver Height	SH4	H1
6	AD6	Gujarat	Rajkot	Jamnagar Road	COPPER CITY	CC5	HN60
7	AD7	MP	Lucknow	Sabar Road	HighLights	HG2	HN120
8	AD8	Gujarat	Surat	Surat Highway	Sun Raise	S5	SRE30
9	AD9	Gujarat	Rajkot	Raiya Exchange	Ramananda	R1	R10
10	AD10	MP	Ujjain	Gopalpura Road	Niti	N2	NV28
11	AD11	Gujarat	Rajkot	Hospital Chowk	High Pillars	H3	HP32

pgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

Query Editor Query History

```

330 SELECT * FROM Employee;
331 SELECT * FROM Role;
332 SELECT * FROM Customer;
333 SELECT * FROM Orders;
334 SELECT * FROM Payment;
335 SELECT * FROM Product;
336 SELECT * FROM Address;
337 SELECT * FROM Brand;
338 SELECT * FROM Category;
339 SELECT * FROM Completes;
340 SELECT * FROM Have;

```

Data Output Explain Messages Notifications

brand_id	[PK] character varying (20)	bnd_name
1	BR1	Balaji
2	BR2	Everest
3	BR3	Milton
4	BR4	Classmate
5	BR5	Navneet

pgAdmin File Object Tools Help

Browser Dashboard Properties SQL Statistics Dependencies Dependents inventorystoremanagementstore_31.sql

```

330 SELECT * FROM Employee;
331 SELECT * FROM Role;
332 SELECT * FROM Customer;
333 SELECT * FROM Orders;
334 SELECT * FROM Payment;
335 SELECT * FROM Product;
336 SELECT * FROM Address;
337 SELECT * FROM Brand;
338 SELECT * FROM Category;
339 SELECT * FROM Completes;
340 SELECT * FROM Have;

```

Data Output Explain Messages Notifications

categ_id	categ_name
C1	Dry Fruits
C2	Cold Drinks
C3	Food Packets
C4	Stationaries
C5	Milk Products
C6	Pulses
C7	Grains

pgAdmin File Object Tools Help

Browser Dashboard Properties SQL Statistics Dependencies Dependents inventorystoremanagementstore_31.sql

```

330 SELECT * FROM Employee;
331 SELECT * FROM Role;
332 SELECT * FROM Customer;
333 SELECT * FROM Orders;
334 SELECT * FROM Payment;
335 SELECT * FROM Product;
336 SELECT * FROM Address;
337 SELECT * FROM Brand;
338 SELECT * FROM Category;
339 SELECT * FROM Completes;
340 SELECT * FROM Have;

```

Data Output Explain Messages Notifications

completes_id	prod_id	order_id	ordered_quantity
CMP1	PR1	OD1	10
CMP2	PR3	OD1	2
CMP3	PR1	OD2	22
CMP5	PR2	OD4	10

pgAdmin File Object Tools Help

Browser Dashboard Properties SQL Statistics Dependencies Dependents inventorystoremanagementstore_31.sql

```

338 SELECT * FROM Category;
339 SELECT * FROM Completes;
340 SELECT * FROM Have;
341 SELECT * FROM SuppliedBy;
342 SELECT * FROM Supplier;
343 SELECT * FROM Supplies;
344 --EXTRA QUERIES:
345 --SEEING WHICH SUPPLIER HAVE dealing with brand 1
346
347 --SELECT sup_id FROM SuppliedBy WHERE brand_id = 'BR1';
348 SELECT sup_id FROM SuppliedBy WHERE brand_id = 'BR1';

```

Data Output Explain Messages Notifications

have_id	categ_id	brand_id
H11	C1	BR1
H12	C4	BR3
H13	C5	BR2
H14	C2	BR4
H15	C6	BR5

pgAdmin File Object Tools Help

Browser Dashboard Properties SQL Statistics Dependencies Dependents inventorystoremanagementstore_31.sql

```

338 SELECT * FROM Category;
339 SELECT * FROM Completes;
340 SELECT * FROM Have;
341 SELECT * FROM SuppliedBy;
342 SELECT * FROM Supplier;
343 SELECT * FROM Supplies;
344
345 --EXTRA QUERIES:
346
347 --SEEING WHICH SUPPLIER HAVE dealing with brand 1
348 SELECT sup_id FROM SuppliedBy WHERE brand_id = 'BR1';

```

Data Output Explain Messages Notifications

suppliedby_id	brand_id	sup_id
SB1	BR1	SU1
SB2	BR2	SU1
SB3	BR1	SU3
SB4	BR5	SU2
SB5	BR4	SU3
SB6	BR3	SU1

pgAdmin File Object Tools Help

Browser Dashboard Properties SQL Statistics Dependencies Dependents inventorystoremanagementstore_31.sql

```

338 SELECT * FROM Category;
339 SELECT * FROM Completes;
340 SELECT * FROM Have;
341 SELECT * FROM SuppliedBy;
342 SELECT * FROM Supplier;
343 SELECT * FROM Supplies;
344
345 --EXTRA QUERIES:
346
347 --SEEING WHICH SUPPLIER HAVE dealing with brand 1
348 SELECT sup_id FROM SuppliedBy WHERE brand_id = 'BR1';

```

Data Output Explain Messages Notifications

sup_id	first_name	last_name	phone_1	phone_2
SU1	Ravindar	Sharma	7624584621	8426137942
SU2	Dhya	Chauhan	9826475234	[null]
SU3	Kamlesh	Bhatt	964251732	8426137942

pgAdmin File Object Tools Help

Browser Dashboard Properties SQL Statistics Dependencies Dependents inventorystoremanagementstore_31.sql

```

338 SELECT * FROM Category;
339 SELECT * FROM Completes;
340 SELECT * FROM Have;
341 SELECT * FROM SuppliedBy;
342 SELECT * FROM Supplier;
343 SELECT * FROM Supplies;
344
345 --EXTRA QUERIES:
346
347 --SEEING WHICH SUPPLIER HAVE dealing with brand 1
348 SELECT sup_id FROM SuppliedBy WHERE brand_id = 'BR1';

```

Data Output Explain Messages Notifications

supplies_id	prod_id	sup_id	supplies_quantity	total_price
SP1	PR1	SU1	20	50
SP2	PR3	SU1	2	30
SP3	PR1	SU3	30	60
SP4	PR2	SU2	50	20

QUERIES:

PgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

- > Casts
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Schemas (1)
 - > public
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > Aa FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > 1. Sequences
 - > Tables (14)
 - > address
 - > brand
 - > category
 - > completes
 - > customer
 - > employee
 - > have

Query Editor Query History

```

344
345 --EXTRA QUERIES:
346
347 --SEEING WHICH SUPPLIER HAVE dealing with brand 1
348 SELECT sup_id FROM SuppliedBy WHERE brand_id = 'BR1';
349
350 --SEEING FOR ORDER WHAT'S THE TOTAL PRICE FOR COMPLETED ONE
351 SELECT total_price,order_id FROM Orders WHERE completed = 'YES';
352
353 -- SEEING WHICH OF THE ORDER NEED TO BE COMPLETED
354 SELECT order_id FROM Orders WHERE completed = 'NO';
355
356 --SEEING DISTINCT ID FOR BRAND via HAVE TABLE
357 SELECT DISTINCT brand_id FROM HAVE;
358
359 --SEEING LAST ORDER MADE BY CUSTOMER 2 as we want to see the last order first we can use
360 --order by while using descending order on the date limiting it to 1 so as to show only the top most row

```

Data Output Explain Messages Notifications

sup_id	character varying(20)
1	SU1
2	SU3

PgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

- > Casts
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Schemas (1)
 - > public
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > Aa FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > 1. Sequences
 - > Tables (14)
 - > address
 - > brand
 - > category

Query Editor Query History

```

344
345 --EXTRA QUERIES:
346
347 --SEEING WHICH SUPPLIER HAVE dealing with brand 1
348 SELECT sup_id FROM SuppliedBy WHERE brand_id = 'BR1';
349
350 --SEEING FOR ORDER WHAT'S THE TOTAL PRICE FOR COMPLETED ONE
351 SELECT total_price,order_id FROM Orders WHERE completed = 'YES';
352
353 -- SEEING WHICH OF THE ORDER NEED TO BE COMPLETED
354 SELECT order_id FROM Orders WHERE completed = 'NO';
355
356 --SEEING DISTINCT ID FOR BRAND via HAVE TABLE

```

Data Output Explain Messages Notifications

total_price	order_id
1	440 002
2	50 003
3	220 001
4	50 004

PgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

- > Casts
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Schemas (1)
 - > public
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > Aa FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > 1. Sequences
 - > Tables (14)
 - > address

Query Editor Query History

```

344
345 --SEEING WHICH SUPPLIER HAVE dealing with brand 1
346 SELECT sup_id FROM SuppliedBy WHERE brand_id = 'BR1';
347
348 --SEEING FOR ORDER WHAT'S THE TOTAL PRICE FOR COMPLETED ONE
349 SELECT total_price,order_id FROM Orders WHERE completed = 'YES';
350
351 -- SEEING WHICH OF THE ORDER NEED TO BE COMPLETED
352 SELECT order_id FROM Orders WHERE completed = 'NO';
353
354 --SEEING DISTINCT ID FOR BRAND via HAVE TABLE
355 SELECT DISTINCT brand_id FROM HAVE;
356
357 --SEEING LAST ORDER MADE BY CUSTOMER 2 as we want to see the last order first we can use
358 --order by while using descending order on the date limiting it to 1 so as to show only the top most row

```

Data Output Explain Messages Notifications

order_id	character varying(20)
1	005

PgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

- > Casts
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Schemas (1)
 - > public
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > Aa FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > 1. Sequences
 - > Tables (14)
 - > address
 - > brand
 - > category

Query Editor Query History

```

344
345 --SEEING FOR ORDER WHAT'S THE TOTAL PRICE FOR COMPLETED ONE
346 SELECT total_price,order_id FROM Orders WHERE completed = 'YES';
347
348 -- SEEING WHICH OF THE ORDER NEED TO BE COMPLETED
349 SELECT order_id FROM Orders WHERE completed = 'NO';
350
351 --SEEING DISTINCT ID FOR BRAND via HAVE TABLE
352 SELECT DISTINCT brand_id FROM HAVE;
353
354 --SEEING LAST ORDER MADE BY CUSTOMER 2 as we want to see the last order first we can use
355 --order by while using descending order on the date limiting it to 1 so as to show only the top most row

```

Data Output Explain Messages Notifications

brand_id	character varying(20)
1	BR1
2	BR2
3	BR3
4	BR4
5	BR5

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure under 'Inventorystoremanagementsyst'. The main area shows a query editor with the following SQL code:

```
SEEING WHICH OF THE ORDER NEED TO BE COMPLETED  
354 SELECT order_id FROM Orders WHERE completed = 'NO';  
355  
--SEEING DISTINCT ID FOR BRAND via HAVE TABLE  
356 SELECT DISTINCT brand_id FROM HAVE;  
357  
--SEEING LAST ORDER MADE BY CUSTOMER 2 as we want to see the last order first we can use  
358 --order by while using descending order on the date limiting it to 1 so as to show only the top most row  
359  
360 SELECT * FROM Orders WHERE cust_id = 'CU2' ORDER BY date_order DESC LIMIT 1;  
361  
362  
363 --TO SEARCH THE EMPLOYEE WHOSE NAME STARTS WITH A
```

The results of the last query are shown in a table:

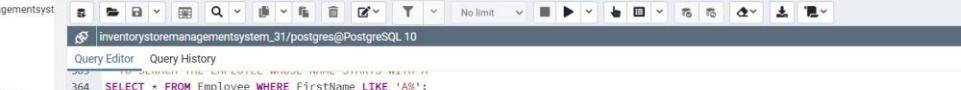
order_id	cust_id	add_id	date_order	total_price	completed	paid
005	CU2	AD7	2021-05-22	0	NO	NO

The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** File, Object, Tools, Help.
- Left Panel (Browser):** Shows the database structure:
 - Inventorystoremanagementsyst (selected)
 - Casts
 - Catalogs
 - Event Triggers
 - Extensions
 - Foreign Data Wrappers
 - Languages
 - Schemas (1)
 - public
 - Collations
 - Domains
 - FTS Configurations
 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Sequences
 - Tables (14)
- Query Editor:** Contains the following SQL code:

```
SELECT DISTINCT brand_id FROM HAVE;  
--SEEING LAST ORDER MADE BY CUSTOMER 2 as we want to see the last order first we can use  
--order by while using descending order on the date limiting it to 1 so as to show only the top most row  
SELECT * FROM Orders WHERE cust_id = 'CU2' ORDER BY date_order DESC LIMIT 1;  
--TO SERACH THE EMPLOYEE WHOSE NAME STARTS WITH A  
SELECT * FROM Employee WHERE FirstName LIKE 'A%';  
--SELECTING THE MAXIMUM SALARY FROM THE EMPLOYEE NAME AND -ROLE FROM ROLE TABLE
```
- Data Output:** Displays the results of the last query:

emp_id	date_of_join	firstname	lastname	salary	age	phone_1	phone_2	role_id
E1	2020-09-16	Ayush	Sharma	25000	26	7826541253	8526971594	R2



The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** File ▾, Object ▾, Tools ▾, Help ▾.
- Header:** inventorystoremanagementstore_31.sql
- Left Sidebar:** Browser, inventorystoremanagementsyst, Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Schemas (1), public, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Tables, Foreign Tables, Functions, Materialized Views, 1.3 Sequences.
- Query Editor:** Contains the following SQL code:

```
364 SELECT * FROM Employee WHERE FirstName LIKE 'A%';
365
366 --SELECTING THE MAXIMUM SALARY FROM THE EMPLOYEE  name AND  ROLE FROM ROLE TABLE
367 SELECT MAX(SALARY) FROM Employee;
368 SELECT Role.role_name,Employee.FirstName FROM Role,Employee
369     WHERE Role.role_id = (SELECT Role_id FROM Employee WHERE Salary = (SELECT MAX(SALARY) FROM Employee))
370     AND Employee.emp_id = (SELECT emp_id FROM Employee WHERE Salary = (SELECT MAX(SALARY) FROM Employee));
371
372 --SHOWING THE PRODUCTS COMING FROM CATEGORY C2 SHOWING IT'S CATEGORY
373 SELECT Product.product_name, ProductCategory.ID, Category.Category_name FROM Product,Categories WHERE Category.ID = "C2"
```
- Data Output:** Shows a table with columns role_name, firstname, and lastname. The first row is Manager, Prakash, and the last row is Salesperson, John.
- Bottom Status Bar:** Shows the current connection as inventorystoremanagementsystem_31/postgres@PostgreSQL 10.

The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** File, Object, Tools, Help.
- Browser:** inventorystoremanagementsyst (selected), Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Schemas (1), public, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views.
- Query Editor:** Query History, inventorystoremanagementsystem_31/postgres@PostgreSQL 10.
The code in the editor is:

```
369 WHERE Role.role_id = (SELECT Role_id FROM Employee WHERE Salary = (SELECT MAX(SALARY) FROM Employee))
370 and Employee.emp_id = (SELECT emp_id FROM Employee WHERE Salary = (SELECT MAX(SALARY) FROM Employee));
371
372 --SHOWING THE PRODUCTS COMING FROM CATEGORY C2 SHOWING IT'S CATEGORY
373 SELECT Product.prod_name, Product.prod_id, Category.categ_name FROM Product,Category WHERE Category.categ_id = 'C2'
374 and Product.prod_id = (SELECT prod_id FROM Product WHERE categ_id = 'C2');
375
376 --ordering products as per the category id
377 SELECT * FROM Product ORDER BY CASE WHEN categ_id = 'C3' THEN '1' WHEN categ_id = 'C5' THEN '2' ELSE categ_id END ASC;
378
```
- Data Output:** Shows a table with columns prod_name, prod_id, and categ_name. The data is:

	prod_name	prod_id	categ_name
1	Maaza	PR3	Cold Drinks

pgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

Query Editor Query History

```

373 SELECT Product.prod_name, Product.prod_id, Category.catge_name FROM Product,Category WHERE Category.catge_id = 'C2'
374     and Product.prod_id = (SELECT prod_id FROM Product WHERE catge_id = 'C2');

375
376 --ordering products as per the catge id
377 SELECT * FROM Product ORDER BY CASE WHEN catge_id = 'C3' THEN '1' WHEN catge_id = 'C5' THEN '2' ELSE catge_id END ASC;

378
379 --seeing that for all employee how much does it cost for there salary
380 SELECT SUM(SALARY) FROM Employee;

381
382 --seeing the supplier and address details via inner join

```

Data Output Explain Messages Notifications

prod_id	prod_name	unit_price	prod_quantity	catge_id
PR1	Chocolate	20	68	C5
PR3	Maaza	10	20	C2
PR2	Pen	5	130	C4

pgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

Query Editor Query History

```

370 --seeing the products as per the case
371 SELECT * FROM Product ORDER BY CASE WHEN catge_id = 'C3' THEN '1' WHEN catge_id = 'C5' THEN '2' ELSE catge_id END ASC;

372
373 --seeing that for all employee how much does it cost for there salary
380 SELECT SUM(SALARY) FROM Employee;

381
382 --seeing the supplier and address details via inner join
383 SELECT * FROM Supplier INNER JOIN Address ON Supplier.sup_id = Address.sup_id;

384
385 --seeing customer address via left join
386 SELECT * FROM Customer LEFT JOIN Address ON Customer.cust_id = Address.cust_id;

```

Data Output Explain Messages Notifications

sum	real
1	130000

pgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

Query Editor Query History

```

370 --seeing the products as per the case
371 SELECT * FROM Product ORDER BY CASE WHEN catge_id = 'C3' THEN '1' WHEN catge_id = 'C5' THEN '2' ELSE catge_id END ASC;

372
373 --seeing that for all employee how much does it cost for there salary
380 SELECT SUM(SALARY) FROM Employee;

381
382 --seeing the supplier and address details via inner join
383 SELECT * FROM Supplier INNER JOIN Address ON Supplier.sup_id = Address.sup_id;

384
385 --seeing customer address via left join
386 SELECT * FROM Customer LEFT JOIN Address ON Customer.cust_id = Address.cust_id;

```

Data Output Explain Messages Notifications

sup_id	first_name	last_name	phone_1	phone_2	add_id	state	city
SU1	Ravinder	Sharma	7624584621	8426137942	A9	Gujarat	Rajkot
SU2	Divya	Chauhan	9826475234	[null]	A10	MP	Ujjain
SU3	Karnlesh	Bhatt	964251732	8426137942	A11	Gujarat	Rajkot

pgAdmin File Object Tools Help

Browser inventorystoremanagementsyst

Query Editor Query History

```

380
381
382 --seeing customer address via left join
383 SELECT * FROM Customer LEFT JOIN Address ON Customer.cust_id = Address.cust_id;
384
385 --FULL JOIN TO SEE CUSTOMER AND ORDERS AT A TIME
386 SELECT * FROM Customer FULL JOIN Orders ON Customer.cust_id = Orders.cust_id;
387
388
389 --SELECTING CUSTOMER WHICH ARE ADDEED BY EMPLOYEE 1 AND 3
390
391 --SELECTING CUSTOMER WHICH ARE ADDEED BY EMPLOYEE 1 AND 3
392 SELECT Firstname, LastName, emp_id FROM Customer WHERE emp_id IN ('E1','E3');
393
394

```

Data Output Explain Messages Notifications

cust_id	firstname	lastname	phone_1	phone_2	emp_id	add_id	state
CU1	Aakash	Tripathi	982583751	7852697432	E1	A6	Gujarat
CU2	Lalit	Kriplani	8426873156	[null]	E2	A7	MP
CU3	Suman	Patel	7526438512	7852697432	E4	A8	Gujarat

The screenshot shows the PGAdmin interface with a connection to the database `inventorystoremanagementstore_31`. The query editor contains the following SQL code:

```
384
385 --seeing customer address via left join
386 SELECT * FROM Customer LEFT JOIN Address ON Customer.cust_id = Address.cust_id;
387
388 --FULL JOIN TO SEE CUSTOMER AND ORDERS AT A TIME
389 SELECT * FROM Customer FULL JOIN Orders ON Customer.cust_id = Orders.cust_id;
390
391 --SELECTING CUSTOMER WHICH ARE DEEDED BY EMPLOYEE 1 AND 3
392 SELECT FirstName, LastName, emp_id FROM Customer WHERE emp_id IN ('E1','E3');
393
```

The Data Output tab is selected, showing the results of the last query:

cust_id	firstname	lastname	phone_1	phone_2	emp_id	order_id	cust_id
1 CU2	Lalt	Kriplani	8426873156	[null]	E2	005	CU2
2 CU2	Lalit	Kriplani	8426873156	[null]	E2	002	CU2
3 CU3	Suman	Patel	7526438512	7852697432	E4	003	CU3
4 CU1	Aakash	Tripathi	9825836751	7852697432	E1	001	CU1
5 CU2	Lalit	Kriplani	8426873156	[null]	E2	004	CU2

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure under 'Inventorystoremanagementsyst'. The main area has a toolbar at the top with various icons. Below the toolbar is a search bar and a dropdown menu. The central part contains a 'Query Editor' tab with the following SQL code:

```
SELECT * FROM Customer FULL JOIN Orders ON Customer.cust_id = Orders.cust_id;  
--SELECTING CUSTOMER WHICH ARE ADDED BY EMPLOYEE 1 AND 3  
SELECT FirstName, LastName, emp_id FROM Customer WHERE emp_id IN ('E1','E3');  
--SELECTING EMPLOYEE ID AND FIRSTNAME WHOSE AGE BETWEEN 26 TO 30  
SELECT emp_id, FirstName, age FROM Employee WHERE age BETWEEN 26 and 30;  
--indexing on employee  
CREATE INDEX emp_idx ON Employee(emp_id);
```

Below the code editor, there are tabs for 'Data Output', 'Explain', 'Messages', and 'Notifications'. A results table is shown for the first query:

	firstname	lastname	emp_id
1	Aakash	Tripathi	E1

The screenshot shows the PGAdmin 4 interface. The left sidebar displays the database structure under 'inventorystoremanagementsyst'. The main area shows a query editor with the following SQL code:

```
388 -- FULL JOIN TO SEE CUSTOMER AND ORDERS AT A TIME
389 SELECT * FROM Customer FULL JOIN Orders ON Customer.cust_id = Orders.cust_id;
390
391 --SELECTING CUSTOMER WHICH ARE ADDED BY EMPLOYEE 1 AND 3
392 SELECT FirstName, LastName,emp_id FROM Customer WHERE emp_id IN ('E1','E3');
393
394 -- SELECTING EMPLOYEE ID AND FIRSTNAME WHOSE AGE BETWEEN 26 TO 30
395 SELECT emp_id, FirstName,age FROM Employee WHERE age BETWEEN 26 and 30;
396
397 --indexing on employee
398 CREATE INDEX employee_idx ON Employee(emp_id);
```

Below the query editor, there are tabs for Data Output, Explain, Messages, and Notifications. The Data Output tab is selected, showing a table with the following data:

emp_id	firstname	age
E1	Ayush	26
E4	Neelam	29

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with various objects like Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Schemas, and more. The main area is a query editor titled 'inventorystoremanagementstore_31.sql' which contains the following SQL code:

```
--indexing on employee
CREATE INDEX empind ON Employee(emp_id);

--CREATING VIEW FOR CUSTOMER OF ITS ADDRESS SO AS TO SHOW ONLY LIMITED VIEW TO CUSTOMER
CREATE VIEW showcustadd AS SELECT Customer.cust_id, Customer.FirstName, Address.state, Address.city FROM Customer,Address
    WHERE Customer.cust_id = Address.cust_id;

SELECT * FROM showcustadd;
```

Below the code, the 'Messages' tab shows the result: 'CREATE INDEX'. At the bottom, a message states 'Query returned successfully in 60 msec.'

```

PgAdmin File Object Tools Help
Browser Dashboard Properties SQL Statistics Dependencies Dependents inventorystoremanagementstore_31.sql
inventorystoremanagementsystem_31/postgres@PostgreSQL 10
Query Editor Query History
396 --indexing on employee
398 CREATE INDEX empind ON Employee(emp_id);
399
400 --CREATING VIEW FOR CUSTOMER OF ITS ADDRESS SO AS TO SHOW ONLY LIMITED VIEW TO CUSTOMER
401 CREATE VIEW showcustadd AS SELECT Customer.cust_id, Customer.FirstName, Address.state, Address.city FROM Customer,Address
402 WHERE Customer.cust_id = Address.cust_id;
403
404 SELECT * FROM showcustadd;
405
Data Output Explain Messages Notifications
cust_id character varying (20) firstname character varying (30) state character varying (15) city character varying (15)
1 CU1 Aakash Gujarat Rajkot
2 CU2 Lalit MP Lucknow
3 CU3 Suman Gujarat Surat

```

→ Conclusion:

From the above project I did for the inventory store management system I began with the understanding of inventory store, what may be the general requirement, how can be things be differentiated and all the basic root cause. I have tried meeting every aspect as much as possibility. I learnt that how database can be designed which are the different queries that runs behind when we do some activity. I have written some the queries such that manually no one has to each and every time update the table as the product quantity is supplied, instead of that I have wrote a query which only takes the supply id and will fetch the information that which product is supplied and in which quantity and will increase in total product quantity. Same way I have learnt many other queries running behind while giving order, making payment, observing if the order is completed or not and so on. By the above project I learnt how to approach the project, build step by step the ideation and how to implement and update as per requirement.