Amaze-On - Database Analysis and Design Project

INDENG 215

Aakash Sundaresan

Ankit Nitinkumar Bhawsar

Apurva Arni

Arnaud Minondo

Jack O'Donoghue

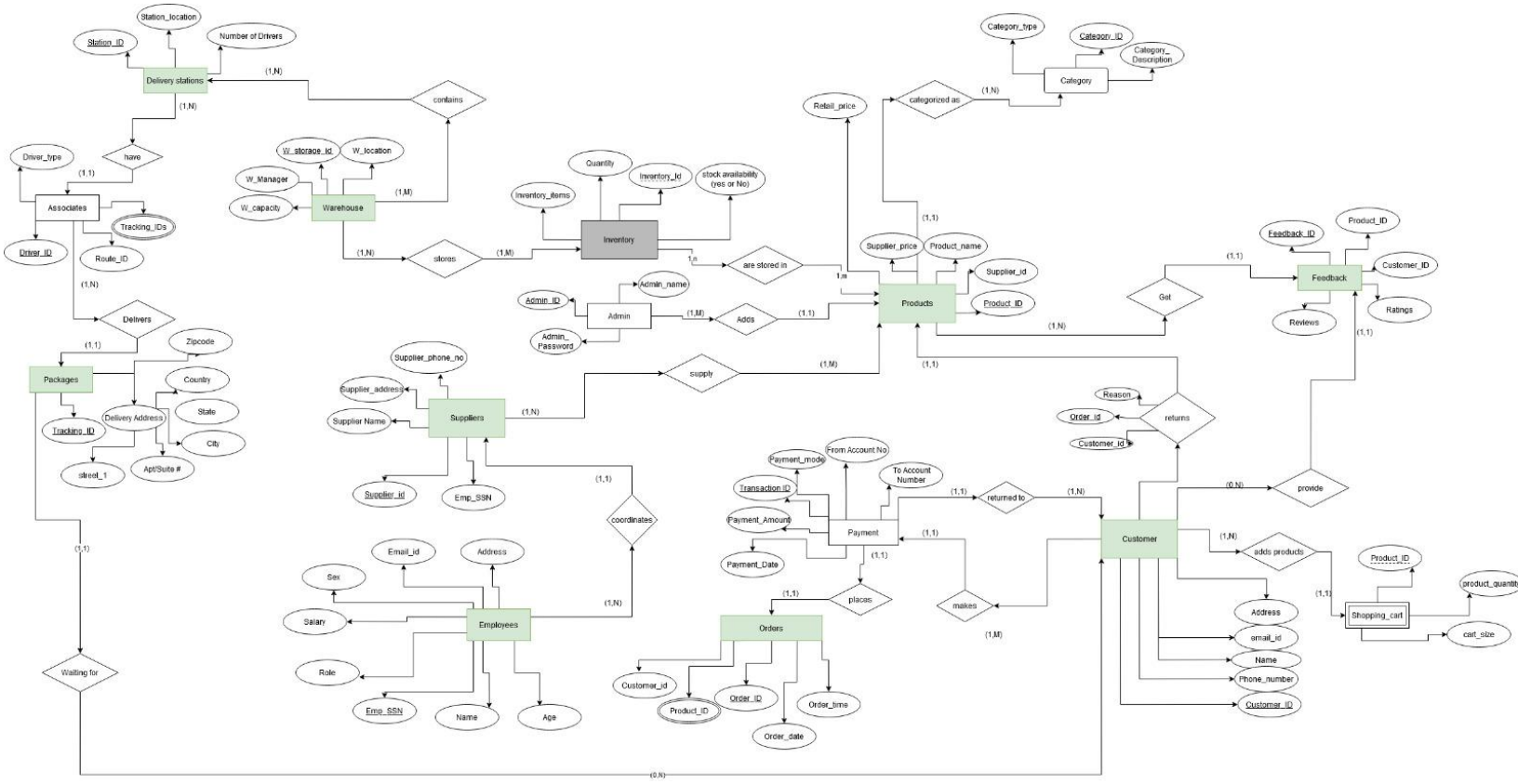
Serah Varghese

Shravani Nimbolkar

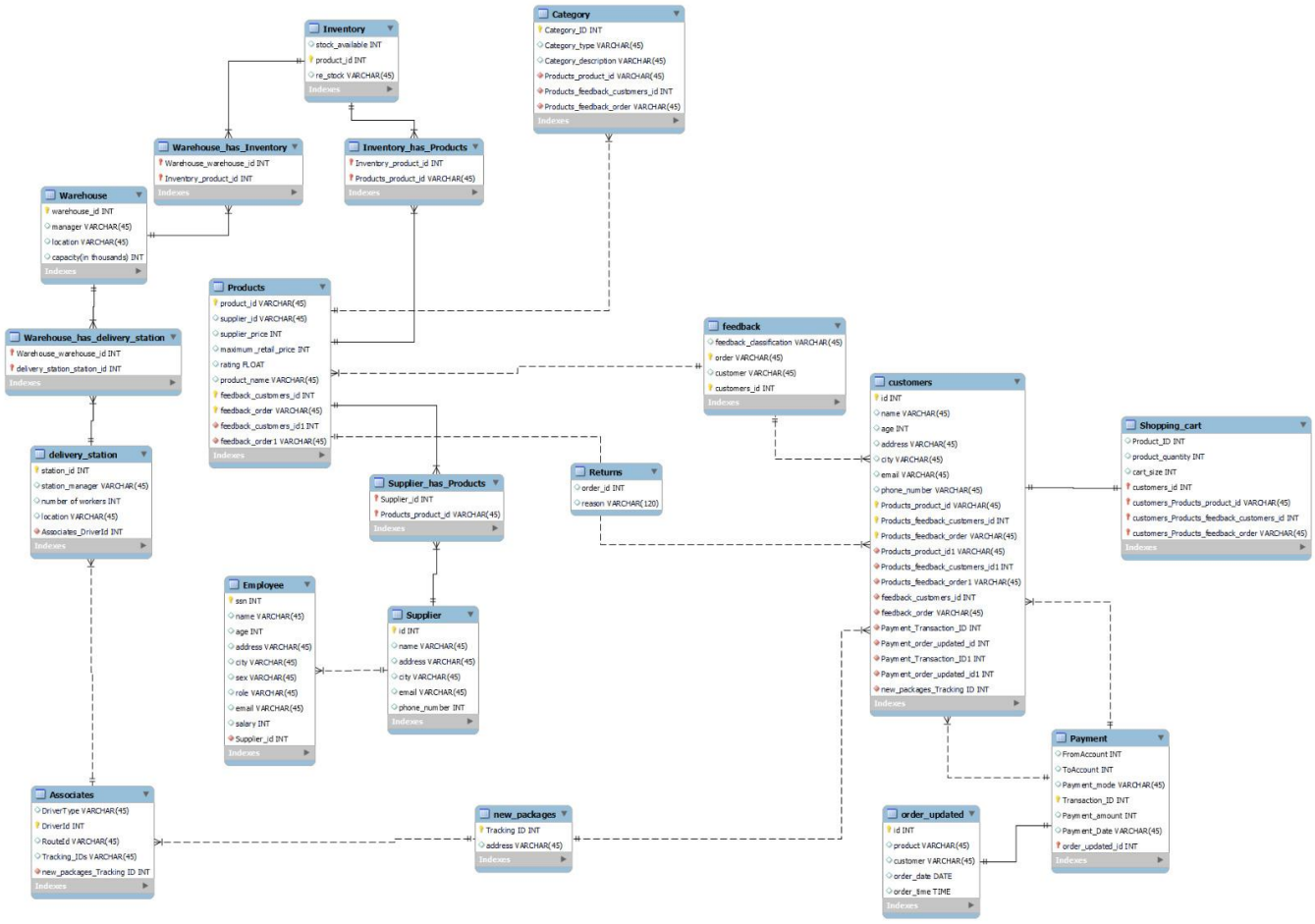
Overview

Amaze-On is an e-commerce company which majorly operates in the United States. They work with suppliers in procuring products which they deliver to their customers. Due to the high volume of data that accompanies running such a company, Amaze-On maintains databases for each and every one of their domains.

Entity Relationship



Schema(Workbench)



Assumptions

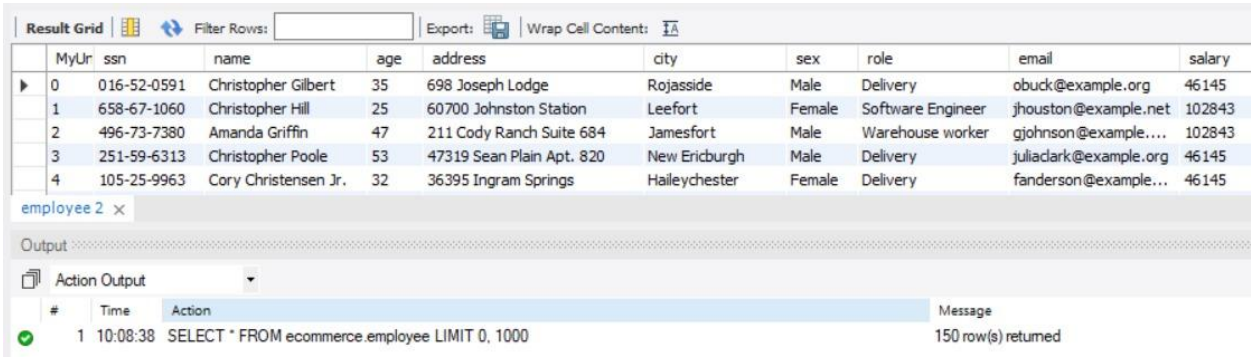
- Each associate represents only one delivery station but each delivery station can have multiple associates.
- Every feedback is associated with one customer only.
- Each supplier must coordinate with one employee.However, one employee can manage multiple suppliers.
- A new product can only be added into the system by an Admin. One Admin can add multiple products.
- All payments from/to the customer can only be associated with one customer.

Data

Once we constructed the EER diagram, we moved on to generate the data. For this purpose. We used the pydbgen and faker packages on python and generated data for all the entities in our schema. Snippets of code are shown below.

```
import pydbgen
from pydbgen import pydbgen
myDB = pydbgen.pydb()
from faker import Faker
fake = Faker()
employee = pd.DataFrame(columns = ['ssn','name','age','address','city','sex','role','email'])
for i in range(150):
    #phone_employee = fake.phone_number()
    city_employee = fake.city()
    name_employee = fake.name()
    address_employee = fake.street_address()
    ssn_employee = fake.ssn()
    age_employee = random.randint(21,60)
    role_employee = random.choice(("Data Scientist", "Software Engineer","Product Manager","Sales",
                                   "Quality Control","Delivery","Warehouse worker"))
    email_employee = fake.email()
    sex_employee = random.choice(("Male","Female"))
    a= [ssn_employee,name_employee,age_employee,address_employee,city_employee,sex_employee,role_employee,
          email_employee]
    employee.loc[len(employee.index)] = a
employee["salary"] = np.where(employee["role"] == 'Sales',random.randint(90000,120000),
                              np.where(employee['role'] == 'Data Scientist',random.randint(150000,200000),
                                       np.where(employee["role"] == 'Product Manager',random.randint(150000,225000),
                                                np.where(employee["role"] == 'Delivery',random.randint(25000,50000),
                                                         np.where(employee["role"] == 'Quality Control',random.randint(150000,225000)
                                                                  random.randint(100000,125000))))))
employee["salary"] = employee["salary"].astype("float")
employee
```

Using this we were able to generate the data for the **EMPLOYEE** entity:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	MyUr	ssn	name	age	address	city	sex	role	email	salary
▶	0	016-52-0591	Christopher Gilbert	35	698 Joseph Lodge	Rojasside	Male	Delivery	obuck@example.org	46145
	1	658-67-1060	Christopher Hill	25	60700 Johnston Station	Leefort	Female	Software Engineer	jhouston@example.net	102843
	2	496-73-7380	Amanda Griffin	47	211 Cody Ranch Suite 684	Jamesfort	Male	Warehouse worker	gjohnson@example....	102843
	3	251-59-6313	Christopher Poole	53	47319 Sean Plain Apt. 820	New Ericburgh	Male	Delivery	juliaclark@example.org	46145
	4	105-25-9963	Cory Christensen Jr.	32	36395 Ingram Springs	Haileychester	Female	Delivery	fanderson@example...	46145

employee 2 ×

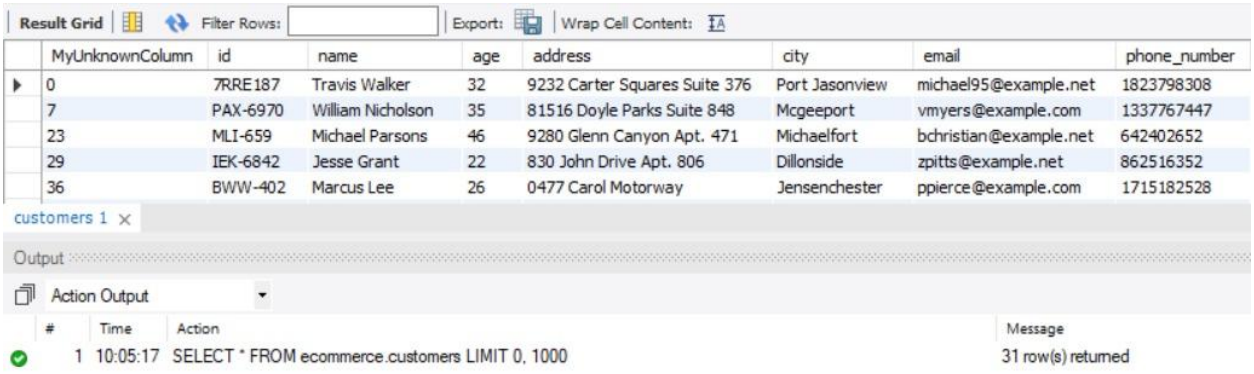
Output

Action Output

#	Time	Action	Message
1	10:08:38	SELECT * FROM ecommerce.employee LIMIT 0, 1000	150 row(s) returned

Similarly, we generated data for all the entities:

2. CUSTOMER:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MyUnknownColumn	id	name	age	address	city	email	phone_number
0	7RRE187	Travis Walker	32	9232 Carter Squares Suite 376	Port Jasonview	michael95@example.net	1823798308
7	PAX-6970	William Nicholson	35	81516 Doyle Parks Suite 848	Mcgeeport	vmyers@example.com	1337767447
23	MLI-659	Michael Parsons	46	9280 Glenn Canyon Apt. 471	Michaelfort	bchristian@example.net	642402652
29	IEK-6842	Jesse Grant	22	830 John Drive Apt. 806	Dillonside	zpitts@example.net	862516352
36	BWW-402	Marcus Lee	26	0477 Carol Motorway	Jensenchester	ppierce@example.com	1715182528

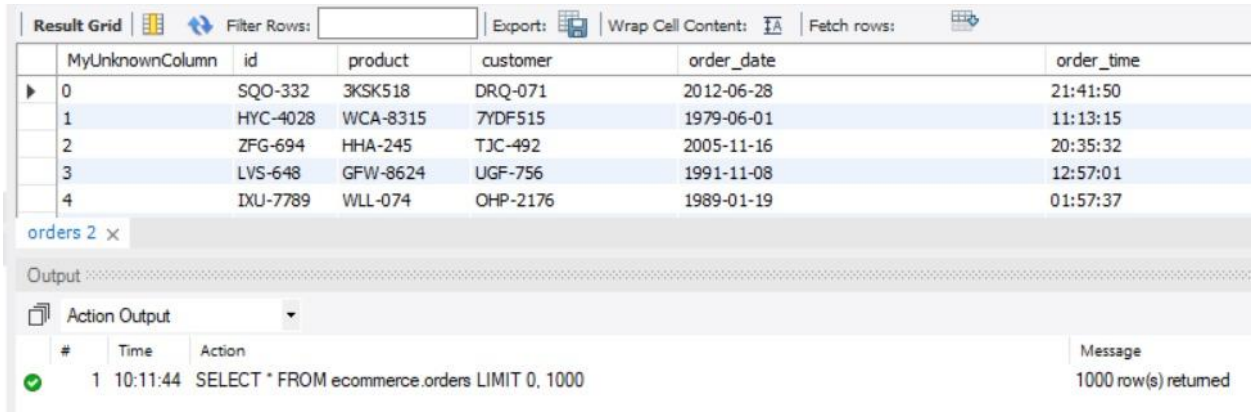
customers 1 ×

Output

Action Output

#	Time	Action	Message
1	10:05:17	SELECT * FROM ecommerce.customers LIMIT 0, 1000	31 row(s) returned

3. ORDERS:



Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

MyUnknownColumn	id	product	customer	order_date	order_time
0	SQO-332	3KSK518	DRQ-071	2012-06-28	21:41:50
1	HYC-4028	WCA-8315	7YDF515	1979-06-01	11:13:15
2	ZFG-694	HHA-245	TJC-492	2005-11-16	20:35:32
3	LVS-648	GFW-8624	UGF-756	1991-11-08	12:57:01
4	IXU-7789	WLL-074	OHP-2176	1989-01-19	01:57:37

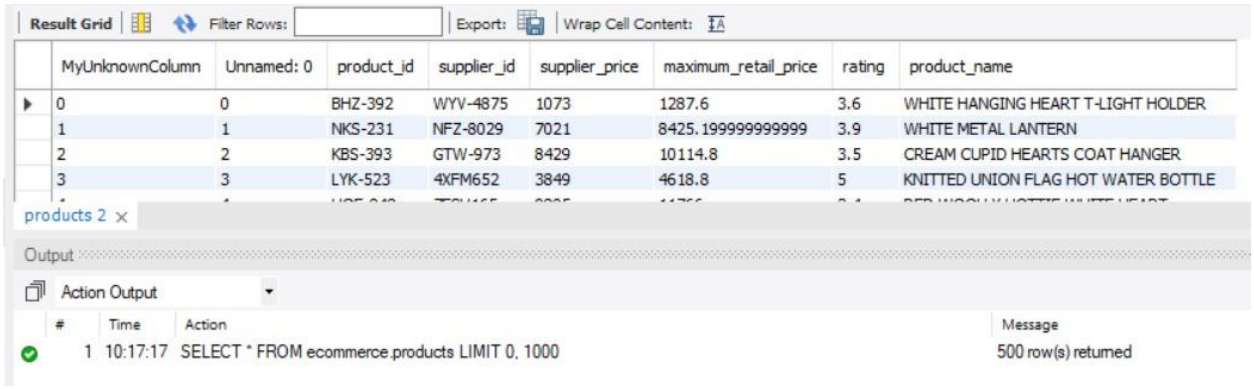
orders 2 ×

Output

Action Output

#	Time	Action	Message
1	10:11:44	SELECT * FROM ecommerce.orders LIMIT 0, 1000	1000 row(s) returned

4. PRODUCTS:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MyUnknownColumn	Unnamed: 0	product_id	supplier_id	supplier_price	maximum_retail_price	rating	product_name
0	0	BHZ-392	WYV-4875	1073	1287.6	3.6	WHITE HANGING HEART T-LIGHT HOLDER
1	1	NKS-231	NFZ-8029	7021	8425.199999999999	3.9	WHITE METAL LANTERN
2	2	KBS-393	GTW-973	8429	10114.8	3.5	CREAM CUPID HEARTS COAT HANGER
3	3	LYK-523	4XFM652	3849	4618.8	5	KNITTED UNION FLAG HOT WATER BOTTLE

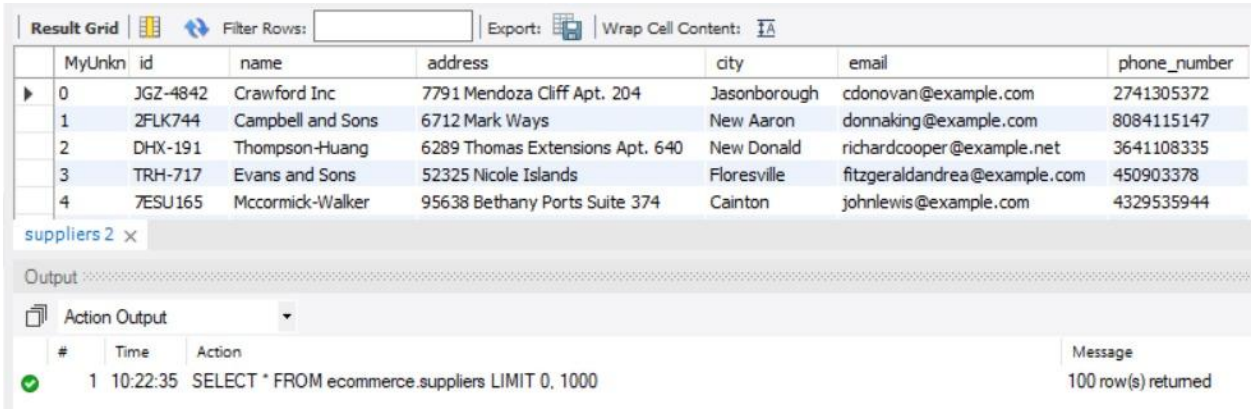
products 2 ×

Output

Action Output

#	Time	Action	Message
1	10:17:17	SELECT * FROM ecommerce.products LIMIT 0, 1000	500 row(s) returned

5. SUPPLIERS:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MyUnkn	id	name	address	city	email	phone_number
0	JGZ-4842	Crawford Inc	7791 Mendoza Cliff Apt. 204	Jasonborough	cdonovan@example.com	2741305372
1	2FLK744	Campbell and Sons	6712 Mark Ways	New Aaron	donnaking@example.com	8084115147
2	DHX-191	Thompson-Huang	6289 Thomas Extensions Apt. 640	New Donald	richardcooper@example.net	3641108335
3	TRH-717	Evans and Sons	52325 Nicole Islands	Floresville	fitzgeraldandrea@example.com	450903378
4	7ESU165	Mccormick-Walker	95638 Bethany Ports Suite 374	Cainton	johnlewis@example.com	4329535944

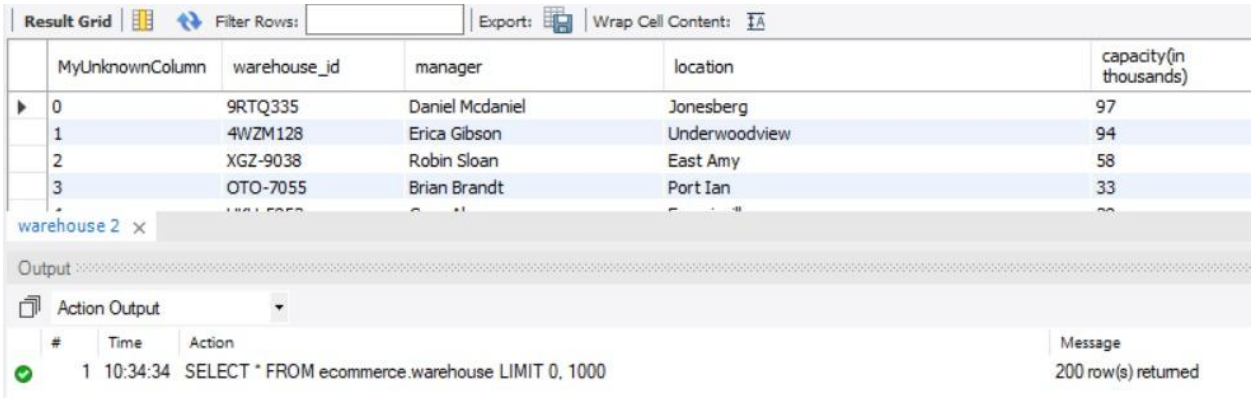
suppliers 2 ×

Output

Action Output

#	Time	Action	Message
1	10:22:35	SELECT * FROM ecommerce.suppliers LIMIT 0, 1000	100 row(s) returned

6. WAREHOUSES:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MyUnknownColumn	warehouse_id	manager	location	capacity(in thousands)
0	9RTQ335	Daniel Mcdaniel	Jonesberg	97
1	4WZM128	Erica Gibson	Underwoodview	94
2	XGZ-9038	Robin Sloan	East Amy	58
3	OTO-7055	Brian Brandt	Port Ian	33

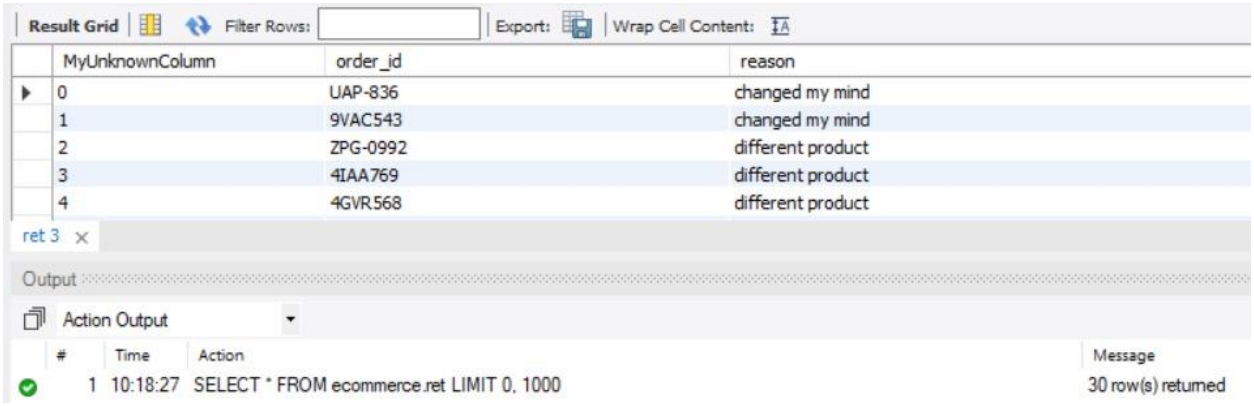
warehouse 2 ×

Output

Action Output

#	Time	Action	Message
1	10:34:34	SELECT * FROM ecommerce.warehouse LIMIT 0, 1000	200 row(s) returned

7. RETURNS:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MyUnknownColumn	order_id	reason
0	UAP-836	changed my mind
1	9VAC543	changed my mind
2	ZPG-0992	different product
3	4IAA769	different product
4	4GVR568	different product

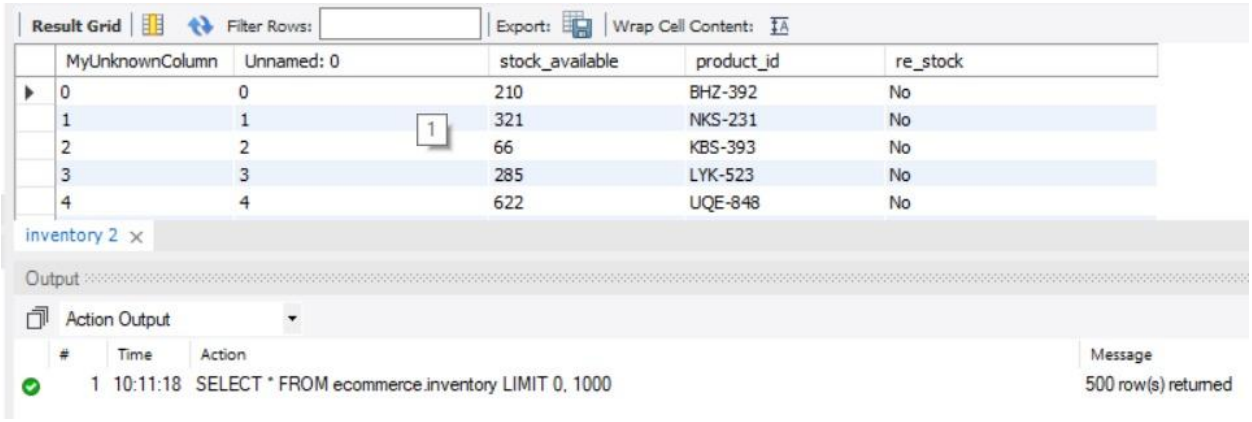
ret 3 ×

Output

Action Output

#	Time	Action	Message
1	10:18:27	SELECT * FROM ecommerce.ret LIMIT 0, 1000	30 row(s) returned

8. INVENTORY:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MyUnknownColumn	Unnamed: 0	stock_available	product_id	re_stock
0	0	210	BHZ-392	No
1	1	321	NKS-231	No
2	2	66	KBS-393	No
3	3	285	LYK-523	No
4	4	622	UQE-848	No

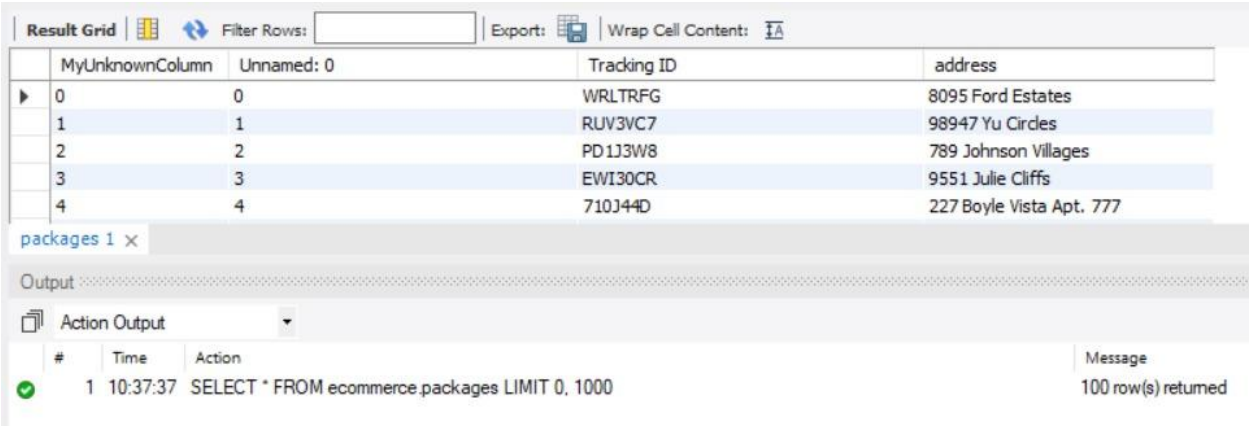
inventory 2 ×

Output

Action Output

#	Time	Action	Message
1	10:11:18	SELECT * FROM ecommerce.inventory LIMIT 0, 1000	500 row(s) returned

9. PACKAGES:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MyUnknownColumn	Unnamed: 0	Tracking ID	address
0	0	WRLTRFG	8095 Ford Estates
1	1	RUV3VC7	98947 Yu Circles
2	2	PD1J3W8	789 Johnson Villages
3	3	EWI30CR	9551 Julie Cliffs
4	4	710J44D	227 Boyle Vista Apt. 777

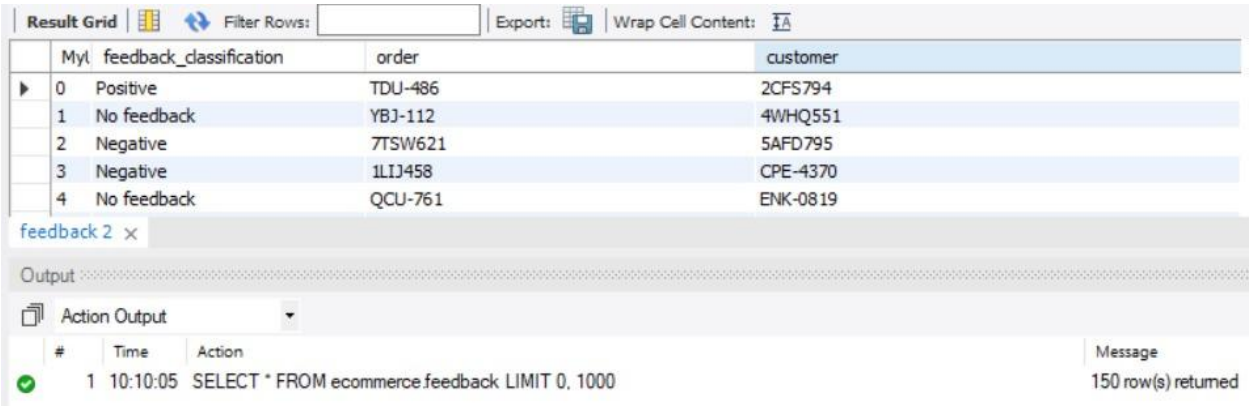
packages 1 ×

Output

Action Output

#	Time	Action	Message
1	10:37:37	SELECT * FROM ecommerce.packages LIMIT 0, 1000	100 row(s) returned

10. FEEDBACKS:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Myl	feedback_classification	order	customer
0	Positive	TDU-486	2CFS794
1	No feedback	YBJ-112	4WHQ551
2	Negative	7TSW621	5AFD795
3	Negative	1LIJ458	CPE-4370
4	No feedback	QCU-761	ENK-0819

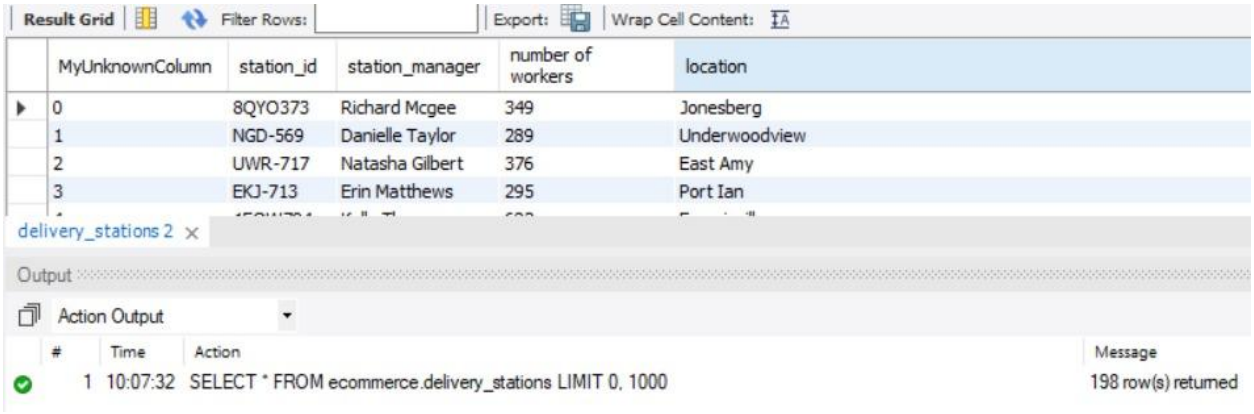
feedback 2 ×

Output

Action Output

#	Time	Action	Message
1	10:10:05	SELECT * FROM ecommerce.feedback LIMIT 0, 1000	150 row(s) returned

11. DELIVERY STATIONS:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MyUnknownColumn	station_id	station_manager	number of workers	location
0	8QYO373	Richard Mcgee	349	Jonesberg
1	NGD-569	Danielle Taylor	289	Underwoodview
2	UWR-717	Natasha Gilbert	376	East Amy
3	EKJ-713	Erin Matthews	295	Port Ian

delivery_stations 2

Output

Action Output

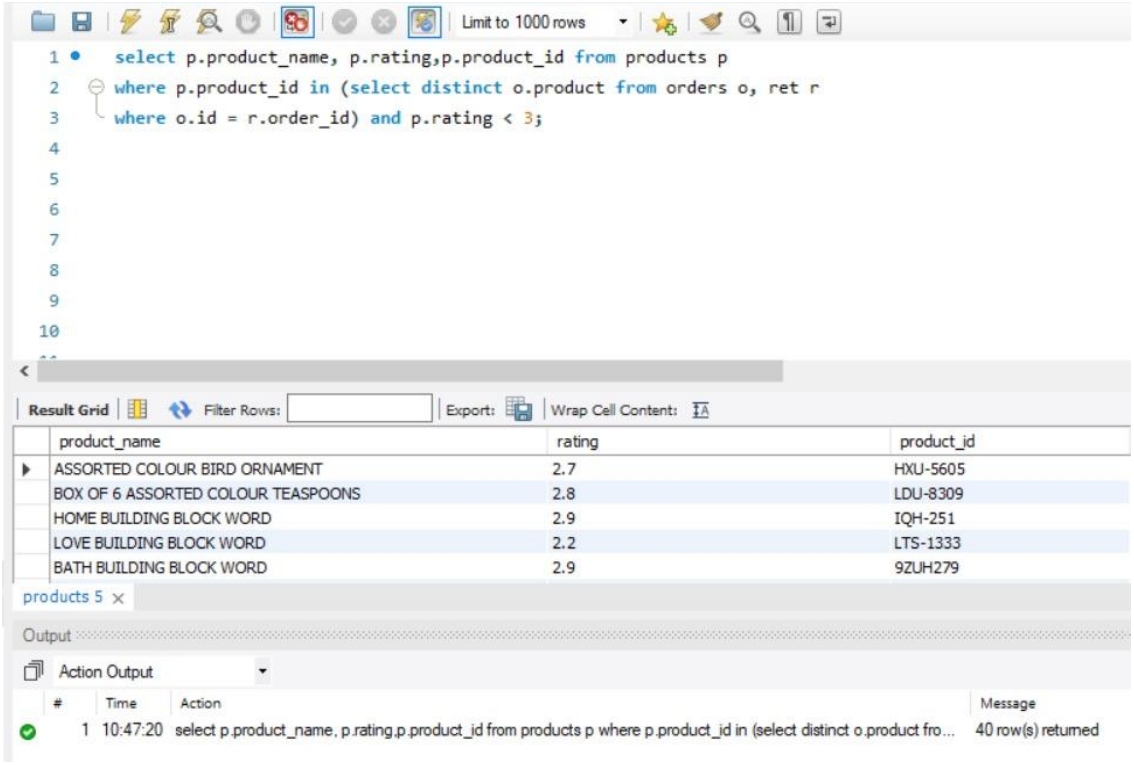
#	Time	Action	Message
1	10:07:32	SELECT * FROM ecommerce.delivery_stations LIMIT 0, 1000	198 row(s) returned

Now that we have created a schema, have generated the data in python and have imported all the datasets in SQL, we can run some queries and perform the data analysis.

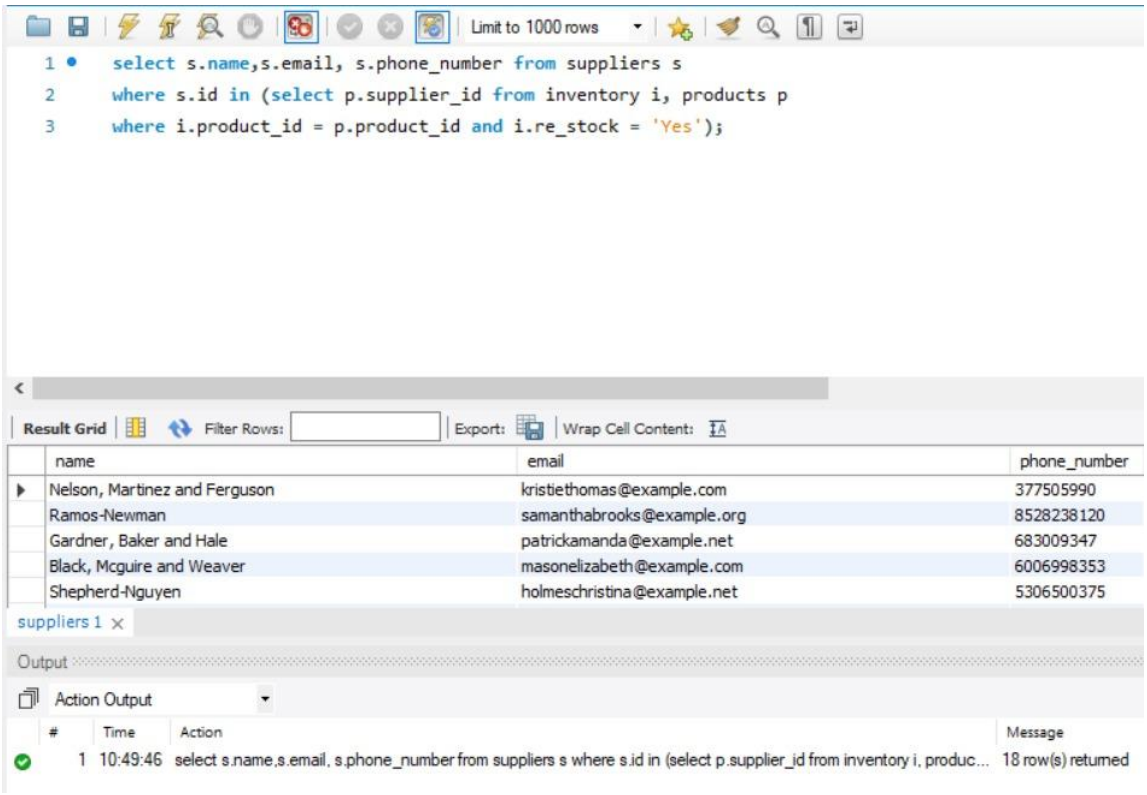
Queries

After we import our dataset in SQL, we run the following queries:

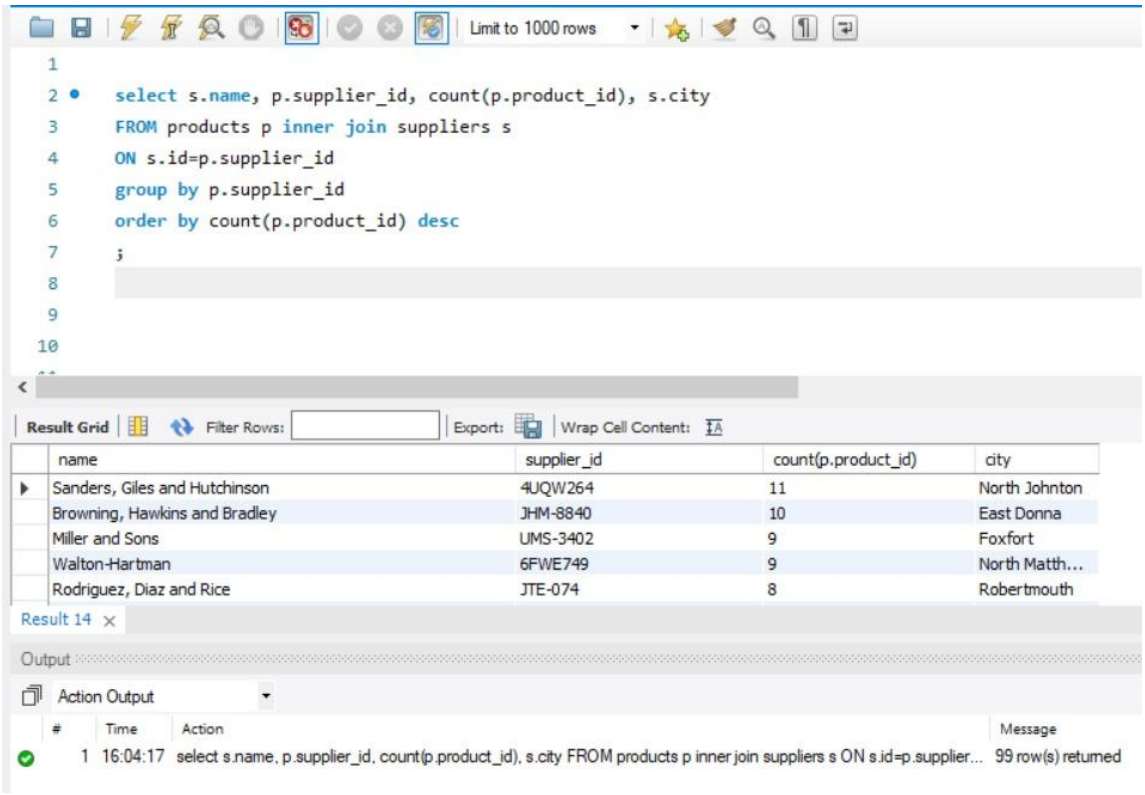
- **Query 1:**
  - **The first query is to determine those products that have been returned and those which have a rating lower than 3. This gives us some insight into quality as, not only have these products been returned they have a very low rating as well. It gives us the name, id and rating of those products.**



- **Query 2:**
  - **The second query provides the details of those suppliers whom we have to contact at this particular moment due to a shortage of items in the inventory which may require restocking. It gives us the name, email id and phone number of the supplier.**



- **Query 3:**
  - **The third query gives us the top suppliers in descending order in terms of number of products supplied. It gives us the name of the supplier name, id city and number of distinct products that they supply.**



```
select s.name, p.supplier_id, count(p.product_id), s.city
FROM products p inner join suppliers s
ON s.id=p.supplier_id
group by p.supplier_id
order by count(p.product_id) desc
;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	supplier_id	count(p.product_id)	city
Sanders, Giles and Hutchinson	4UQW264	11	North Johnton
Browning, Hawkins and Bradley	JHM-8840	10	East Donna
Miller and Sons	UMS-3402	9	Foxfort
Walton-Hartman	6FWE749	9	North Matth...
Rodriguez, Diaz and Rice	JTE-074	8	Robertmouth

Result 14

Output

Action Output

#	Time	Action	Message
1	16:04:17	select s.name, p.supplier_id, count(p.product_id), s.city FROM products p inner join suppliers s ON s.id=p.supplier...	99 row(s) returned

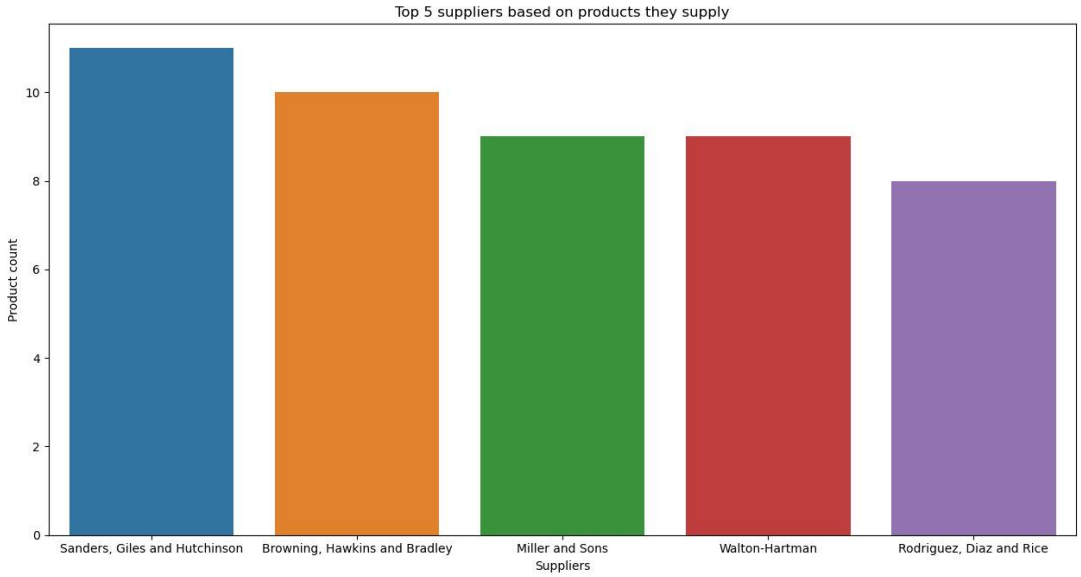
Analysis(Python)

Now that we have run the queries and obtained the results, we can import the result back onto python to perform some analysis.

- **First, we construct a bar chart to show the top 5 suppliers based on the number of products that they supply. The code for the following analysis is:**

```
import mysql.connector
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

mydb=mysql.connector.connect(host="localhost",user="root",password="radha",database="ecommerce")
mycursor=mydb.cursor()
mycursor.execute("select s.name, p.supplier_id, count(p.product_id), s.city FROM products p inner join suppliers s ON s.id=p.supplier_id group by p.supplier_id order b
result = mycursor.fetchall
print(mycursor)

Names = []
product_count = []

for i in mycursor:
    Names.append(i[0])
    product_count.append(i[2])
raw_data = {
'Name' : Names[0:5], 'Count':product_count[0:5]
}


from matplotlib.offsetbox import FontProperties
plt.figure(figsize=(12,15))
ax = sns.barplot( y= 'Count',x = 'Name' , data = raw_data)
# sns.set_context("paper", rc={"font.size":10,"axes.titlesize":10,"axes.labelsize":10})
ax.set_ylabel('Product count', size=10)
ax.set_xlabel('Suppliers', size=10)
ax.set_title('Top 5 suppliers based on #product they supply')


for p in ax.patches:
    ax.annotate(int(p.get_width()),
                ((p.get_x() + p.get_width()),
                 p.get_y()),
                 xytext=(1, -18),
                fontsize=5,
                color='#004d00',
                horizontalalignment='right')
plt.show()
```

By running the following code, we receive the following result:



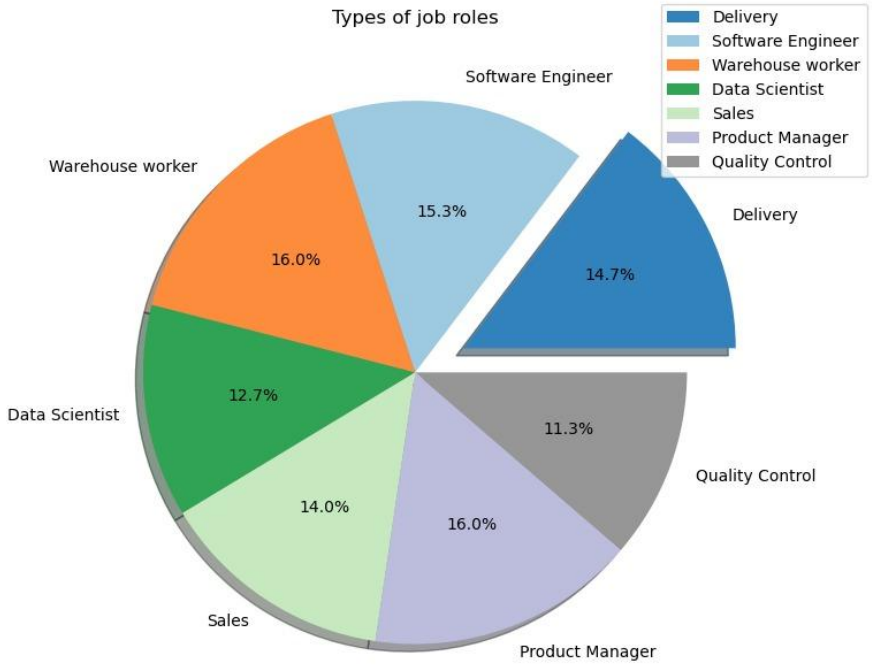
This shows that the top 5 suppliers for the company are Sanders Giles and Hutchinson, Browning Hawkins and Bradley, Miller and Sons, Walton Hartman and Rodriguez Diaz and Rice.

- **Next we construct a pie-chart to determine the different kinds of job roles at Amaze-On. This gives us important insights regarding the composition of employees and whether the company has been hiring based on the required workload i.e. the department with the most workload has adequate number of resources(employees) and also to ensure not a lot of manpower is being wasted on departments with low workloads. The code for the above is:**

top5suppliers.py | roles.py | genderrole.py

```
import mysql.connector
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

mydb=mysql.connector.connect(host="localhost",user="root",password="radha",database="ecommerce")
mycursor=mydb.cursor()
mycursor.execute("SELECT role, count(ssn) from employee group by role;")
result = mycursor.fetchall
print(mycursor)

role = []
count = []

for i in mycursor:
    role.append(i[0])
    count.append(i[1])

'''raw_data = {
'Name' : Names[0:5], 'Count':product_count[0:5]
}'''

mylabels = role
myexplode = [0.2, 0,0,0,0,0,0]
cmap = plt.get_cmap('tab20c')
colors = [cmap(i) for i in np.linspace(0, 1, 8)]

plt.pie(count, labels=mylabels,  explode = myexplode, shadow = True,autopct='%1.1f%%', colors=colors)
plt.legend(bbox_to_anchor=(0.85,1.055), loc="upper left")
plt.title("Types of job roles")
plt.show()

```

After running the following code, we get the following pie chart:



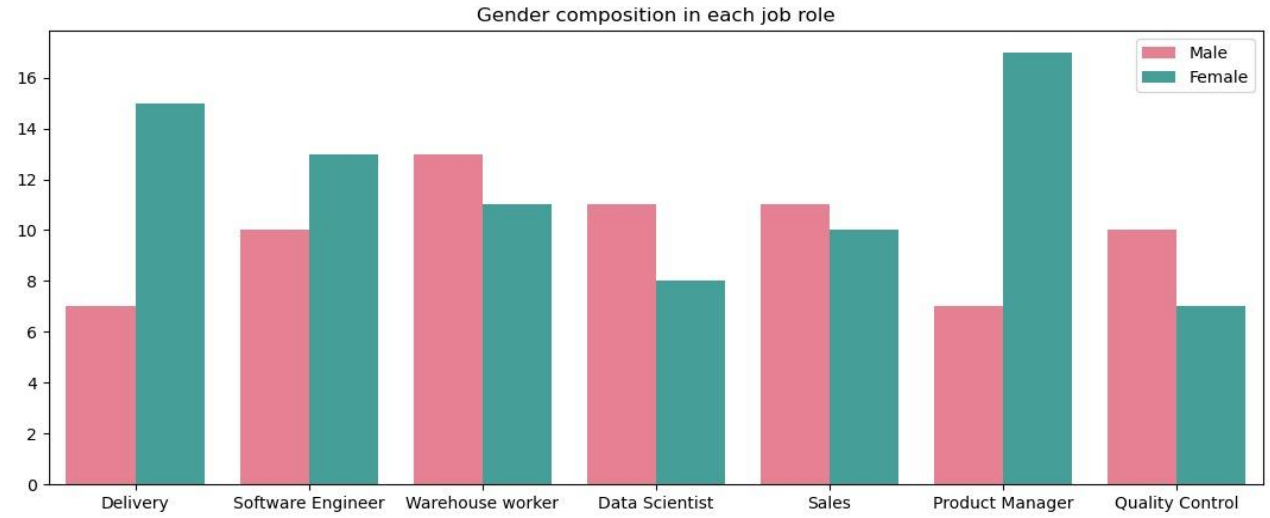
We can conclude from the above pie chart that Product managers and Warehouse workers make up the majority of the employees followed by software engineers, delivery workers and so on. Using this information, management can determine and make decisions accordingly to utilize the right number of resources and the need for hiring or recruiting people wherever needed.

- **This next graph shows us the gender composition in each role within Amaze-On. For that we run the following code:**

genderrole.py

```
import mysql.connector
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

mydb=mysql.connector.connect(host="localhost",user="root",password="radha",database="ecommerce")
mycursor=mydb.cursor()
mycursor.execute("SELECT role, sex, count(ssn) from employee group by role,sex;")
result = mycursor.fetchall
print(mycursor)

role = []
sex = []
count = []

for i in mycursor:
    role.append(i[0])
    sex.append(i[1])
    count.append(i[2])

data = {
'role' : role, 'sex': sex, 'count': count
}
sns.barplot(x="role", y="count", hue="sex", data=data,palette="husl")
plt.title("Gender composition in each job role")
plt.show()
```

After running that we get the following grouped bar graph:



We can see that in a few roles there is domination of one gender over other but predominantly gender equality has been maintained. This can be used by the management to keep a close eye on diversity, equality and inclusion in the organization.

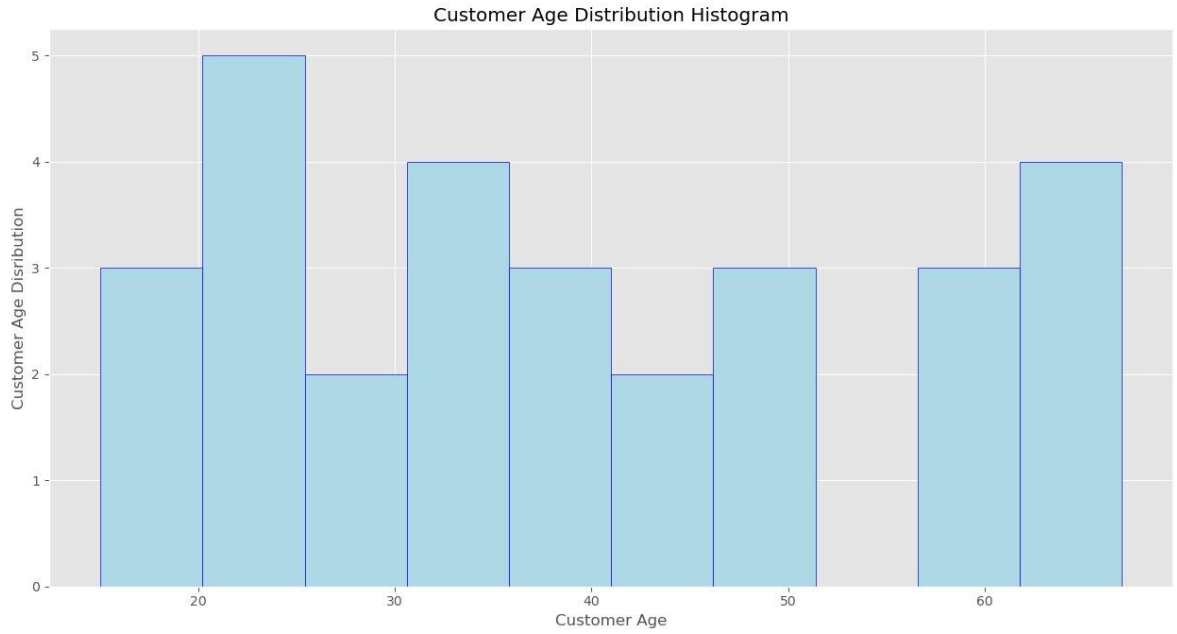
Queries and plots shown above are some of the analyses performed with the data created but the scope of the data is not limited to the above.

- **The next graph indicates what age group our customers are. This is important as in E-commerce understanding the target group for recommendations and customer profiling. This is the code we run for the following:**

genderrole.py | age.py

```
import mysql.connector
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

mydb=mysql.connector.connect(host="localhost",user="root",password="radha",database="ecommerce")
mycursor=mydb.cursor()
mycursor.execute("SELECT customers.age  FROM customers inner join orders on customers.id = orders.customer group by orders.id;")
result = mycursor.fetchall
print(mycursor)

ages = []
for i in mycursor:
    ages.append(i[0])

import matplotlib.pyplot as plt

plt.style.use('ggplot')
plt.hist(ages, color = "lightblue", ec="blue")
plt.xlabel("Customer Age")
plt.ylabel("Customer Age Disribution")
plt.title("Customer Age Distribution Histogram")
plt.show()
```

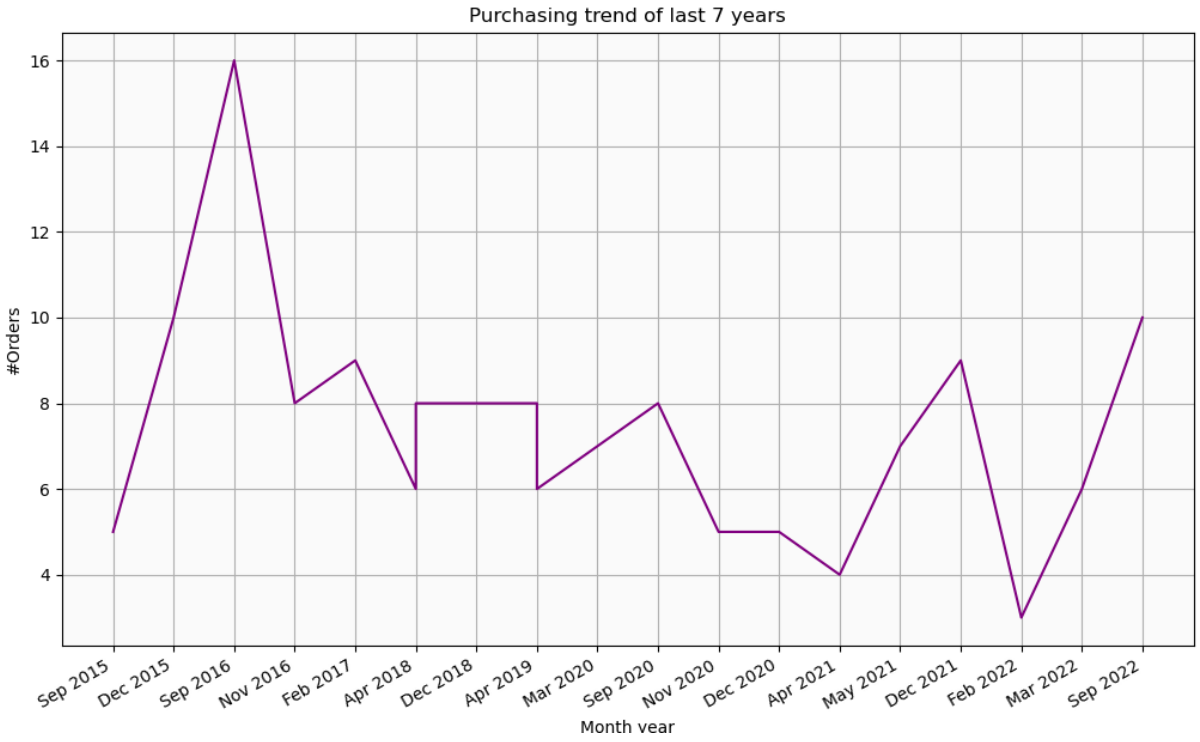
We get the following graph as a result:



Our customer base has people belonging from the age group of 15-30(youngsters), and 30-45(middle age), 45-77(senior citizens) . But in general, there seems to be equal demand from most age groups.

- **Next we came up with a line graph which illustrates the purchasing trend of the customers from the past 7 years from 2015 to 2022. As it can be seen from the graph**

below, the purchasing trend went downhill after the pandemic as the economy and market went down with increasing inflation rates and decreasing purchasing power of customers. Though some signs of recovery can be seen, there are still high fluctuations and the business will take a significant amount of time to bounce back to its normal state.



genderrole.py | age.py | trend.py

```python
import mysql.connector
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

mydb=mysql.connector.connect(host="localhost",user="root",password="radha",database="ecommerce")
mycursor=mydb.cursor()
mycursor.execute("SELECT order_date, year(order_date), MONTHNAME(order_date) AS month, count(order_date)
FROM orders group by id order by order_date asc;")
result = mycursor.fetchall
dates = []
year = []
month = []
orders = []
for i in mycursor:
    dates.append(i[0])
    year.append(i[1])
    month.append(i[2])
    orders.append(i[3])
date = dates[-20:]
year = year[-20:]
month = month[-20:]
order = orders[-20:]
monthyear=[]
for i in range(len(month)):
    monthyear.append(month[i][0:3] + ' ' + str(year[i]))
# Using graph_objects
plt.xticks(rotation=30, ha='right')
plt.plot(monthyear, order, color="purple")
plt.title("Purchasing trend of last 7 years")
plt.xlabel("Month year")
plt.ylabel("#Orders")
ax = plt.axes()
ax.set_facecolor("#FAFAFA")
plt.grid()
plt.show()
```

Conclusion

Using SQL and python, we are able to carry out complex processes like constructing a schema, generating data to populate the entities in the schema, import them into MySQL Workbench to run SQL queries in order to make management decisions and run analysis for the same. Amaze-on can now streamline their whole business model in order to cut down on costs and generate maximum profits by using the analysis carried out by our team. Using these tools we can perform much more specific tasks which are in accordance with the company's needs at that particular point in time.