

Data Science

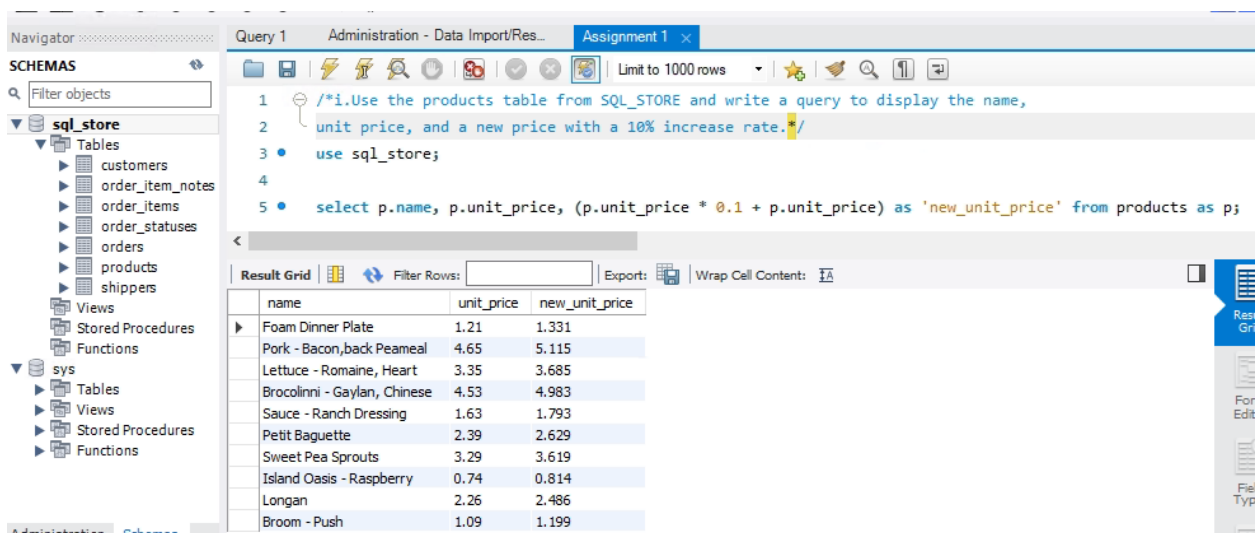
Data Mining Techniques

DQL Queries

Task 1:

DQL Queries

- i. Use the products table from SQL_STORE and write a query to display the name, unit price, and a new price with a 10% increase rate.



The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including 'sql_store' and 'sys' schemas. The 'sql_store' schema is expanded, showing tables like 'customers', 'order_item_notes', 'order_items', 'order_statuses', 'orders', 'products', and 'shippers'. The 'products' table is selected. The main pane displays a SQL query in a text editor, which is executed. The query is as follows:

```
1  /*i.Use the products table from SQL_STORE and write a query to display the name,  
2  unit price, and a new price with a 10% increase rate.*/  
3  use sql_store;  
4  
5  select p.name, p.unit_price, (p.unit_price * 0.1 + p.unit_price) as 'new_unit_price' from products as p;
```

Below the query editor, the 'Result Grid' shows the results of the query. The grid has four columns: 'name', 'unit_price', and 'new_unit_price'. The results are as follows:

name	unit_price	new_unit_price
Foam Dinner Plate	1.21	1.331
Pork - Bacon,back Peameal	4.65	5.115
Lettuce - Romaine, Heart	3.35	3.685
Brocolinni - Gaylan, Chinese	4.53	4.983
Sauce - Ranch Dressing	1.63	1.793
Petit Baguette	2.39	2.629
Sweet Pea Sprouts	3.29	3.619
Island Oasis - Raspberry	0.74	0.814
Longan	2.26	2.486
Broom - Push	1.09	1.199

The round() function can be used to create a unit price with two decimal places. Additionally, this will round decimal places. For instance, the round() function changed the 'Petit Baguette's unit

price from 2.629 to 2.63.

The screenshot shows the SQL Developer interface. The left pane displays the 'SCHEMAS' tree with 'sql_store' expanded. The main query window contains the following SQL code:

```
1  /*i.Use the products table from SQL_STORE and write a query to display the name,  
2  unit price, and a new price with a 10% increase rate.*/  
3  use sql_store;  
4  
5  select p.name, p.unit_price, round((p.unit_price * 0.1 + p.unit_price), 2) as 'new_unit_price'  
6  from products as p;
```

The 'Result Grid' shows the output of the query:

name	unit_price	new_unit_price
Foam Dinner Plate	1.21	1.33
Pork - Bacon,back Peameal	4.65	5.12
Lettuce - Romaine, Heart	3.35	3.69
Brocolinni - Gaylan, Chinese	4.53	4.98
Sauce - Ranch Dressing	1.63	1.79
Petit Baguette	2.39	2.63
Sweet Pea Sprouts	3.29	3.62
Island Oasis - Raspberry	0.74	0.81
Longan	2.26	2.49
Broom - Push	1.09	1.20

It is even feasible to obtain the true two decimal places without rounding by using the truncate() function. The 'Petit Baguette' product had a unit price of 2.629; however, after utilizing the truncate() function, it changed to 2.62.

The screenshot shows the SQL Developer interface. The left pane displays the 'SCHEMAS' tree with 'sql_store' expanded. The main query window contains the following SQL code:

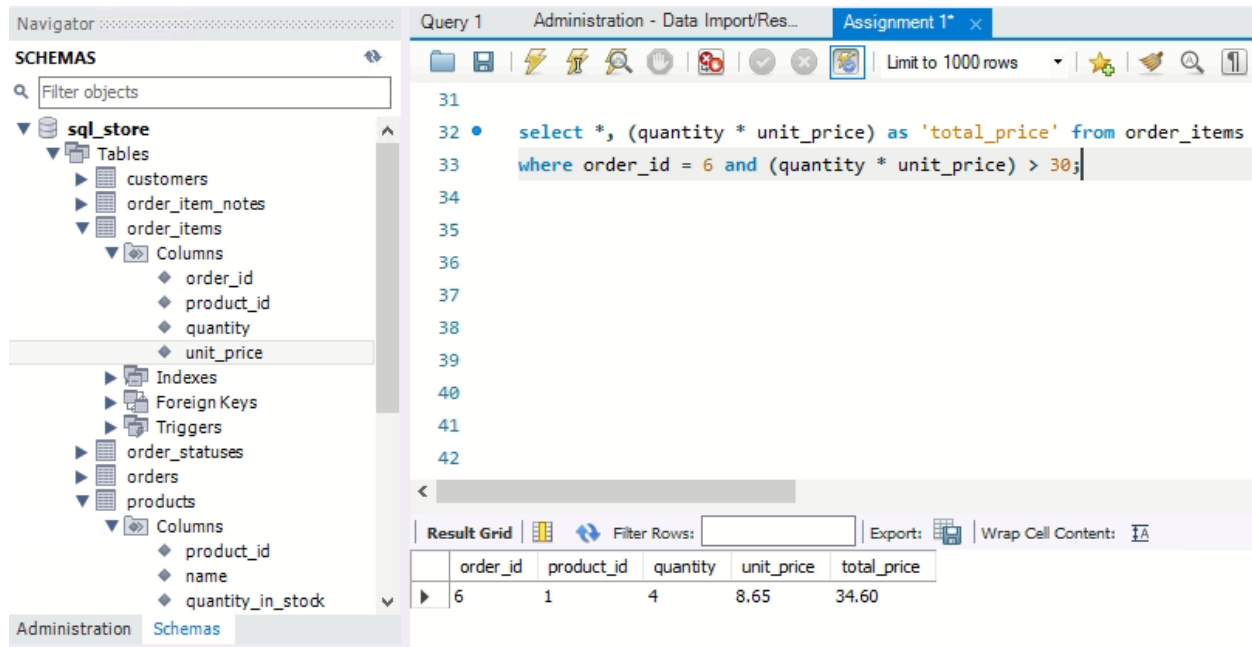
```
8  
9  
10  
11  
12  
13  select p.name, p.unit_price, truncate((p.unit_price * 0.1 + p.unit_price), 2) as 'new_unit_price'  
14  from products as p;
```

The 'Result Grid' shows the output of the query:

name	unit_price	new_unit_price
Foam Dinner Plate	1.21	1.33
Pork - Bacon,back Peameal	4.65	5.11
Lettuce - Romaine, Heart	3.35	3.68
Brocolinni - Gaylan, Chinese	4.53	4.98
Sauce - Ranch Dressing	1.63	1.79
Petit Baguette	2.39	2.62
Sweet Pea Sprouts	3.29	3.61
Island Oasis - Raspberry	0.74	0.81
Longan	2.26	2.48
Broom - Push	1.09	1.19

- ii. Write a query to get the items for order #6 with a total price of over 30. Use the ORDER_ITEMS table from SQL_STORE database.

The order_items table returns four default columns and one additional custom column called total_price when the query below is run.



The screenshot shows a database management interface with a left-hand 'SCHEMAS' pane and a main query editor. The 'SCHEMAS' pane shows the 'sql_store' database with tables like 'customers', 'order_item_notes', 'order_items', 'order_statuses', 'orders', and 'products'. The 'order_items' table is selected, showing columns: 'order_id', 'product_id', 'quantity', and 'unit_price'. The main query editor shows a SQL query: `select *, (quantity * unit_price) as 'total_price' from order_items where order_id = 6 and (quantity * unit_price) > 30;`. Below the query editor is a 'Result Grid' showing one row of data.

order_id	product_id	quantity	unit_price	total_price
6	1	4	8.65	34.60

The query below simply creates and shows the product ID field, which is when the order equals six and the total price is more than thirty.

The screenshot shows the SQL Developer interface. On the left, the 'SCHEMAS' pane displays the 'sql_store' database structure, including tables like 'customers', 'order_item_notes', 'order_items', 'order_statuses', 'orders', and 'products'. The 'order_items' table is expanded, showing columns: 'order_id', 'product_id', 'quantity', and 'unit_price'. The 'products' table is also expanded, showing columns: 'product_id', 'name', and 'quantity_in_stock'. The 'Administration' tab is selected at the bottom.

The 'Query 1' window shows the following SQL query:

```

35
36 • select product_id from order_items
37   where order_id = 6 and (quantity * unit_price) > 30;
38
39
40
41
42
43
44
45
46

```

The 'Result Grid' at the bottom shows the following data:

product_id
1

Due to an inner join between the order_items and products tables, the query below creates six columns. The column names are order id, product id, product name, order quantity, product unit price, and total order price.

The screenshot shows the SQL Developer interface. On the left, the 'SCHEMAS' pane displays the 'sql_store' database structure, including tables like 'customers', 'order_item_notes', 'order_items', 'order_statuses', 'orders', and 'products'. The 'order_items' table is expanded, showing columns: 'order_id', 'product_id', 'quantity', and 'unit_price'. The 'products' table is also expanded, showing columns: 'product_id', 'name', and 'quantity_in_stock'. The 'Administration' tab is selected at the bottom.

The 'Query 1' window shows the following SQL query:

```

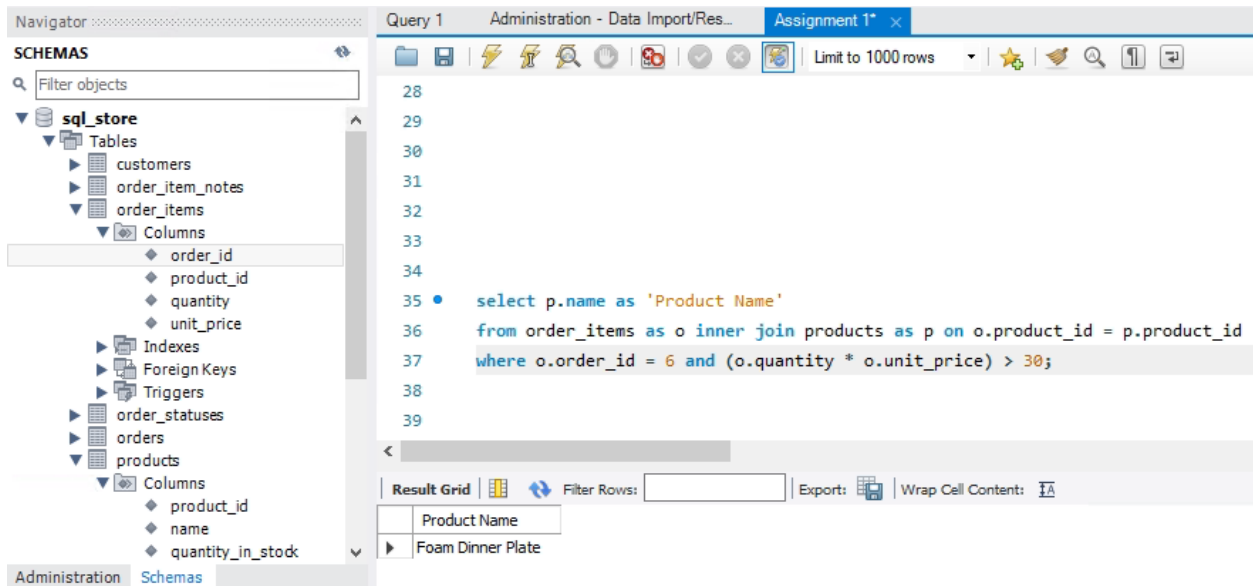
17
18
19
20 /*Write a query to get the items for order #6 with a total price of over 30. Use the
21 ORDER_ITEMS table from SQL_STORE database*/
22
23 • select o.order_id, p.product_id, p.name, o.quantity, o.unit_price,
24   (o.quantity * o.unit_price) as 'total_price'
25 from order_items as o inner join products as p on o.product_id = p.product_id
26 where o.order_id = 6 and (o.quantity * o.unit_price) > 30;
27
28

```

The 'Result Grid' at the bottom shows the following data:

order_id	product_id	name	quantity	unit_price	total_price
6	1	Foam Dinner Plate	4	8.65	34.60

For the criteria mentioned above, the query below just returns the product name.



Inner joins

- iii. Use SQL_STORE database & join the ORDER_ITEMS table with the products table on the PRODUCT_ID column and select the specific columns of your choice.

The query below inner joins the order_items and products tables using the product_id column. It also makes use of aliases to make the column names easier to comprehend.

Navigator

SCHEMAS

Filter objects

- product_id
- quantity
- unit_price
- Indexes
- Foreign Keys
- Triggers
- order_statuses
- orders
- products
 - Columns
 - product_id
 - name
 - quantity_in_stock
 - unit_price
 - Indexes
 - Foreign Keys
 - Triggers
 - shippers
 - Views
 - Stored Procedures

Administration Schemas

Information

Column: **quantity_in_stock**

Definition:

quantity_in_stock int

Query 1 Administration - Data Import/Res... Assignment 1*

Limit to 1000 rows

```

57 /*Use SQL_STORE database & join the ORDER_ITEMS table with the products table
58 on the PRODUCT_ID column and select the specific columns of your choice. */
59
60 • select p.name as 'Product', o.order_id as 'Order Id', o.quantity as 'Order Qty',
61 o.unit_price as 'Price', p.unit_price as 'Cost', p.quantity_in_stock as 'Stock Qty'
62 from order_items as o inner join products as p on o.product_id = p.product_id;

```

Result Grid

	Product	Order Id	Order Qty	Price	Cost	Stock Qty
▶	Foam Dinner Plate	2	2	9.10	1.21	70
	Foam Dinner Plate	6	4	8.65	1.21	70
	Foam Dinner Plate	10	10	6.01	1.21	70
	Pork - Bacon,back Peameal	5	3	9.89	4.65	49
	Pork - Bacon,back Peameal	6	4	3.28	4.65	49
	Lettuce - Romaine, Heart	3	10	9.12	3.35	38
	Lettuce - Romaine, Heart	4	7	6.99	3.35	38
	Lettuce - Romaine, Heart	6	4	7.46	3.35	38
	Lettuce - Romaine, Heart	7	7	9.17	3.35	38
	Brocolinni - Gaylan, Chinese	1	4	3.74	4.53	90
	Brocolinni - Gaylan, Chinese	2	4	1.66	4.53	90
	Sauce - Ranch Dressing	6	1	3.45	1.63	94
	Sauce - Ranch Dressing	8	2	6.94	1.63	94
	Petit Baguette	2	2	2.94	2.39	14
	Petit Baguette	9	5	7.28	2.39	14
	Island Oasis - Rasperry	8	2	8.59	0.74	26

Result 31

- iv. Join products table of SQL_INVENTORY with ORDER_ITEM_NOTES table of SQL_STORE on PRODUCT_ID and select specific columns.

The following query pulls data from the order_item_notes table in the sql_store database and the products table in the sql_inventory database, producing columns for each product that include its name, on-hand quantity, unit price, and notes.

Navigator

SCHEMAS

Filter objects

- unit_price
- Indexes
- Foreign Keys
- Triggers
- Views
- Stored Procedures
- Functions
- sql_store
 - Tables
 - customers
 - order_item_notes
 - Columns
 - note_id
 - order_id
 - product_id
 - note
- Indexes
- Foreign Keys
- Triggers
- order_items

Administration Schemas

Information

Column: **note**

Collation: utf8mb4_0900_ai_ci

Definition: note varchar(255)

Query 1 Assignment 1*

Limit to 1000 rows

```

64
65
66
67
68
69 /*Join products table of SQL_INVENTORY with ORDER_ITEM_NOTES table of
70 SQL_STORE on PRODUCT_ID and select specific columns*/
71
72
73
74 • select p.name as 'product_name', p.quantity_in_stock, p.unit_price, o.note
75 from sql_inventory.products as p inner join sql_store.order_item_notes as o
76 on p.product_id = o.product_id;
77
78

```

Result Grid

product_name	quantity_in_stock	unit_price	note
Pork - Bacon,back Peameal	49	4.65	first note
Brocolinni - Gaylan, Chinese	90	4.53	second note
Foam Dinner Plate	70	1.21	third note
Lettuce - Romaine, Heart	38	3.35	Fourth note
Lettuce - Romaine, Heart	38	3.35	fifth note
Pork - Bacon,back Peameal	49	4.65	sixth note

Joining multiple tables

- Use the 'SQL_INVOICING' database. Join the 'payments' table with the 'clients' table using the 'CLINETTS_ID' column and then join the 'payments' table with the 'PAYMENT_METHODS' table using the 'PAYMENT_METHOD' column.

The below query shows every column that is part of the payments, clients, and payment_methods tables. The SQL query statement utilizes the "use <database name>" sql query execute the database selection instead of including the database name in the query.

Query 1 Assignment 1* payment_methods payments

Limit to 1000 rows

```

84 /*Use the 'SQL_INVOICING' database. Join the 'payments' table with the 'clients'
85 table using the 'CLIENTS_ID' column and then join the 'payments' table with the
86 'PAYMENT_METHODS' table using the 'PAYMENT_METHOD' column.*/
87
88 • use sql_invoicing;
89
90
91 • select * from payments as p inner join clients as c on
92 p.client_id = c.client_id inner join payment_methods as m on
93 m.payment_method_id = p.payment_method_id;
94
95
96
97
98

```

Result Grid

	payment_id	client_id	invoice_id	date	amount	payment_method	client_id	name	address	city	state	phone
1	5	2	2	2019-02-12	8.18	1	5	Topidounge	0863 Farmco Road	Portland	OR	971-888-9129
2	1	6	6	2019-01-03	74.55	1	1	Vinte	3 Nevada Parkway	Syracuse	NY	315-252-7305
3	3	11	11	2019-01-11	0.03	1	3	Yadel	096 Pawling Parkway	San Francisco	CA	415-144-6037
4	5	13	13	2019-01-26	87.44	1	5	Topidounge	0863 Farmco Road	Portland	OR	971-888-9129
5	3	15	15	2019-01-15	80.31	1	3	Yadel	096 Pawling Parkway	San Francisco	CA	415-144-6037
6	3	17	17	2019-01-15	68.10	1	3	Yadel	096 Pawling Parkway	San Francisco	CA	415-144-6037
7	5	18	18	2019-01-08	32.77	1	5	Topidounge	0863 Farmco Road	Portland	OR	971-888-9129
8	5	18	18	2019-01-08	10.00	2	5	Topidounge	0863 Farmco Road	Portland	OR	971-888-9129

Object Info Result 40 x Read Only

The above same three tables with a few chosen columns are displayed in the query results below.

The screenshot shows a database management interface with a 'Navigator' pane on the left, a 'Query 1' editor in the center, and a 'Result Grid' at the bottom. The 'Navigator' pane shows a tree structure of database objects, including 'sql_invoicing' (Tables: clients, invoices, payment_methods), 'sql_store' (Tables: customers), and 'payments'. The 'Query 1' editor contains the following SQL query:

```

94
95
96
97
98 • select p.date, p.amount, c.name, c.address, c.city, c.state, c.phone,
99      m.name as 'payment_method'
100 from payments as p inner join clients as c on
101      p.client_id = c.client_id inner join payment_methods as m on
102      m.payment_method_id = p.payment_method;
103
104
105
106
107
108

```

The 'Result Grid' displays the following data:

date	amount	name	address	city	state	phone	payment_method
2019-02-12	8.18	Topidounge	0863 Farmco Road	Portland	OR	971-888-9129	Credit Card
2019-01-03	74.55	Vinte	3 Nevada Parkway	Syracuse	NY	315-252-7305	Credit Card
2019-01-11	0.03	Yadel	096 Pawling Parkway	San Francisco	CA	415-144-6037	Credit Card
2019-01-26	87.44	Topidounge	0863 Farmco Road	Portland	OR	971-888-9129	Credit Card
2019-01-15	80.31	Yadel	096 Pawling Parkway	San Francisco	CA	415-144-6037	Credit Card
2019-01-15	68.10	Yadel	096 Pawling Parkway	San Francisco	CA	415-144-6037	Credit Card
2019-01-08	32.77	Topidounge	0863 Farmco Road	Portland	OR	971-888-9129	Credit Card
2019-01-08	10.00	Topidounge	0863 Farmco Road	Portland	OR	971-888-9129	Cash

The 'Information' pane at the bottom left shows details for the 'name' column: Collation: utf8mb4_0900_ai_ci, Definition: name varchar(50).

Outer joins

- vi. Write a query to perform left join on orders table and shippers table of SQL_STORE on SHIPPER_ID and get the details of ORDER_ID, CUSTOMER_ID, ORDER_DATE, SHIPPED_DATE & SHIPPERS_NAME.

The orders table and the shippers table are linked using a left outer join in the query below. As a result, all of the rows from the orders table appear in the results table, and the NULL values in the shipped_date column allows to identify the unshipped orders. In this SQL query, the table names have been directly linked to the database name. (db name.table name)

The screenshot shows a database management interface. On the left, the 'SCHEMAS' pane displays a tree view of the 'sql_store' schema, including tables like 'customers', 'order_item_notes', 'order_items', 'order_statuses', 'orders', 'products', 'indexes', 'foreign keys', and 'triggers'. The 'orders' table is selected, showing its columns: 'order_id', 'customer_id', 'order_date', 'status', 'comments', 'shipped_date', and 'shipper_id'. The main pane shows a SQL query in 'Query 1' with the following text:

```

108
109
110 /*Write a query to perform left join on orders table and shippers table
111 of SQL_STORE on SHIPPER_ID and get the details of ORDER_ID, CUSTOMER_ID,
112 ORDER_DATE, SHIPPED_DATE & SHIPPERS_NAME*/
113
114
115 • select o.order_id, o.customer_id, o.order_date, o.shipped_date
116 from sql_store.orders as o left outer join sql_store.shippers as s
117 on o.shipper_id = s.shipper_id;
118
119

```

Below the query, the 'Result Grid' shows the following data:

	order_id	customer_id	order_date	shipped_date
▶	1	6	2019-01-30	NULL
	2	7	2018-08-02	2018-08-03
	3	8	2017-12-01	NULL
	4	2	2017-01-22	NULL
	5	5	2017-08-25	2017-08-26
	6	10	2018-11-18	NULL
	7	2	2018-09-22	2018-09-23
	8	5	2018-06-08	NULL
	9	10	2017-07-05	2017-07-06
	10	6	2018-04-22	2018-04-23

References

- Abld Ali Awan. (2023, Sep). *What is Named Entity Recognition (NER)? Methods, Use Cases, and Challenges*. Retrieved from datacamp: <https://www.datacamp.com/blog/what-is-named-entity-recognition-ner>
- Akash Bokade. (2023, Aug). *NLP workbook + Amazon Review Sentiment analysis*. Retrieved from kaggle: <https://www.kaggle.com/code/akashbokade/nlp-workbook-amazon-review-sentiment-analysis>
- Chetna Khanna. (2021, Feb 10). *Text pre-processing: Stop words removal using different libraries*. Retrieved from Medium: <https://towardsdatascience.com/text-pre-processing-stop-words-removal-using-different-libraries-f20bac19929a>

Kevin Arvai. (2020, Jul 20). *K-Means Clustering in Python: A Practical Guide*. Retrieved from realpython: <https://realpython.com/k-means-clustering-python/>

Kevin Arvai. (n.d.). *K-Means Clustering in Python: A Practical Guide*. Retrieved from realpython: <https://realpython.com/k-means-clustering-python/>

Martin Porter. (2006, Jan). *The Porter Stemming Algorithm*. Retrieved from tartarus: <https://www.tartarus.org/~martin/PorterStemmer/>

Neri. (2023, Nov 03). *How To Implement Intent Classification In NLP [7 ML & DL Models] With Python Example*. Retrieved from spotintelligence: <https://spotintelligence.com/2023/11/03/intent-classification-nlp/>

Prateek Majumder. (2023, Sep 13). *Named Entity Recognition (NER) in Python with Spacy*. Retrieved from analyticsvidhya: <https://www.analyticsvidhya.com/blog/2021/06/nlp-application-named-entity-recognition-ner-in-python-with-spacy/>

Sadrach Pierre & Matthew Urwin. (2017, Oct 14). *builtin*. Retrieved from How to Form Clusters in Python: Data Clustering Methods: <https://builtin.com/data-science/data-clustering-python>

What is named entity recognition? (n.d.). Retrieved from ibm: <https://www.ibm.com/topics/named-entity-recognition>