

Data Science

Data Mining Techniques

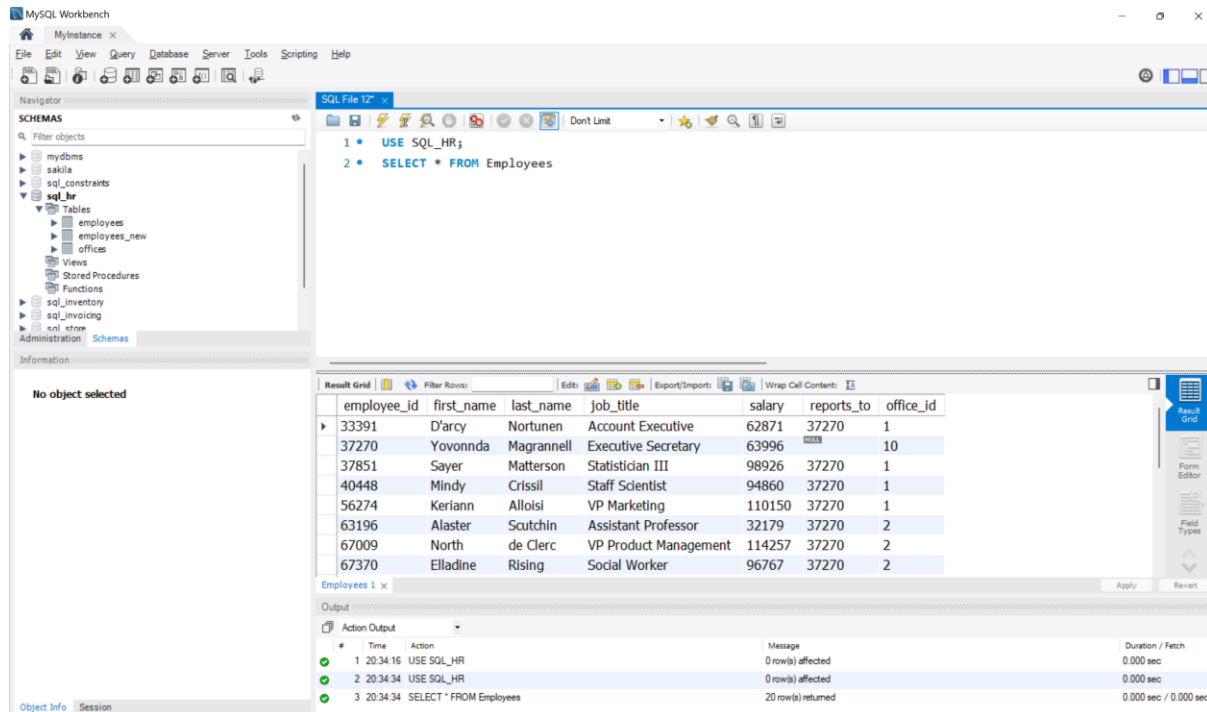
DDL Queries

Task 2:

Guidelines:

1. Use the appropriate database and table mentioned in the exercise to get the correct solution.
2. You need to download and install MySQL workbench.

Sample MySQL Workbench:



3. You need to run the database scripts given, to create databases to work with the queries.
4. Query solution screenshots must be pasted in the same Word file.

DDL

- i. Create a database with the name MYDBMS and create a 'PERSONS' table under MYDBMS database with the following columns.

PERSON_ID,
FIRSTNAME,
LASTNAME,

AGE.

Make 'PERSON_ID' as primary key and give the appropriate data type to each column.

Data types of the table "persons":

Column Name	Data type	Feature
person_id	Int(11)	The column can take, and show a value exceeding the length 11, but these values will not be prefixed with 0s
firstname	varchar(15)	The column allocates dynamically up to 16 bytes, (character strings) up to 15 for data and, at least, 1 additional byte to store the length of the data.
lastname	varchar(25)	The column allocates dynamically up to 26 bytes, (character strings) up to 25 for data and, at least, 1 additional byte to store the length of the data.
age	Int(3)	The column can take, and show a value exceeding the length 3, but these values will not be prefixed with 0s

A new database called mydbms and a table named "persons" are created with the following query.

The screenshot displays a database management interface with three main panels:

- Navigator (Left):** Shows a tree view of schemas. Under the 'mydbms' schema, the 'persons' table is selected. Other schemas like 'sql_inventory' and 'sql_invoicing' are also visible.
- Query Editor (Top Right):** Contains SQL code for creating and querying the 'persons' table.


```

123 • use mydbms;
124
125 • CREATE TABLE `persons` (
126     `person_id` int(11),
127     `firstname` varchar(15),
128     `lastname` varchar(25),
129     `age` int(3),
130     PRIMARY KEY (`person_id`)
131 );
132
133
134
135 • select * from persons;
136
      
```
- Table Information (Bottom Left):** Provides details for the selected 'persons' table.

Table: persons

Columns:

Column Name	Data Type	Constraints
person_id	int	PK
firstname	varchar(15)	
lastname	varchar(25)	
age	int	
- Result Grid (Bottom Right):** Shows the output of the 'select * from persons;' query. It displays a single row with all columns set to 'NULL'.

person_id	firstname	lastname	age
NULL	NULL	NULL	NULL

By applying the `auto_increment` key word in the MySQL query, it is also possible to use the auto increment feature on the `person_id` column. The `person_id` does not need to be manually inserted in that scenario. The only drawback is that this auto increment functionality won't help if the user needs to keep his own numbering scheme because the `person_id` might not always be in sequence mode. However, the user can still insert id numbers as they consider by utilizing the column name (`person_id`) in the insert sql query.

Furthermore, as per the below sql query, null values are not permitted in the “`person_id`” column.

The screenshot displays a MySQL database management interface. On the left, the 'SCHEMAS' pane shows a tree view of the database structure. Under 'mydbms', the 'persons' table is selected, showing its columns: 'person_id', 'firstname', 'lastname', and 'age'. Below this, the 'Information' pane provides details for the 'persons' table, including its columns and their data types and constraints.

The main query editor shows the following SQL code:

```

5
6 • create table `persons` (
7   `person_id` int(6) not null auto_increment,
8   `firstname` varchar(15),
9   `lastname` varchar(25),
10  `age` int(3),
11  primary key (`person_id`));
12
13 • select * from persons;
14
15
16
17
18

```

Below the query editor, the 'Result Grid' shows the results of the query. The grid has four columns: 'person_id', 'firstname', 'lastname', and 'age'. The first row shows all values as 'NULL'.

person_id	firstname	lastname	age
NULL	NULL	NULL	NULL

The 'Information' pane for the 'persons' table shows the following columns:

- person_id**: int AI PK
- firstname**: varchar(15)
- lastname**: varchar(25)
- age**: int

In addition, column constraints can be explicitly defined by using the "default null" key words to indicate that the column can contain null values by default. If not, the MySQL query management system will automatically define it. Therefore, the column constraint must be defined as "not null" when data is required.

Navigator

SCHEMAS

Filter objects

- mydbms
 - Tables
 - persons
 - Columns
 - person_id
 - firstname
 - lastname
 - age
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
 - sql_inventory
 - Tables
 - products
 - Views
 - Stored Procedures
 - Functions

Administration Schemas

Information

Table: **persons**

Columns:

<u>person_id</u>	int AI PK
firstname	varchar(15)
lastname	varchar(25)
age	int

Query 1 Assignment 1* SQL File 2* x

Limit to 1000 rows

```

117
118 • create table `persons` (
119     `person_id` int not null auto_increment,
120     `firstname` varchar(15) default null,
121     `lastname` varchar(25) default null,
122     `age` int default null,
123     primary key (`person_id`)
124 )
125
126 select * from persons;
127
128
129
130

```

Result Grid

	person_id	firstname	lastname	age
*	NULL	NULL	NULL	NULL

persons 25 v

- ii. Insert the following data into the 'PERSONS' created table.

```
-- Person_ID      FirstName      LastName      Age
-- 1              Sachin          Tendulkar      50
-- 2              Taimur         Khan           10
-- 3              Brad           Pitt           55
-- 4              John           Wick           53
-- 5              James          Bond           47
-- 6              Yuraj          Singh          43
```

Inserting values without using column names,

The screenshot shows the SQL Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows the 'mydbms' database with a 'persons' table. The 'Query 1' window contains the following SQL code:

```
19 • INSERT INTO `persons` VALUES (1,'Sachin','Tendulkar',50);
20 • INSERT INTO `persons` VALUES (2,'Taimur','Khan',10);
21 • INSERT INTO `persons` VALUES (3,'Brad','Pitt',55);
22 • INSERT INTO `persons` VALUES (4,'John','Wick',53);
23 • INSERT INTO `persons` VALUES (5,'James','Bond',47);
24 • INSERT INTO `persons` VALUES (6,'Yuraj','Singh',43);
25
26
27 • select * from persons;
28
29
30
31
32
```

Below the query window, the 'Result Grid' shows the data returned by the SELECT statement:

	person_id	firstname	lastname	age
▶	1	Sachin	Tendulkar	50
	2	Taimur	Khan	10
	3	Brad	Pitt	55
	4	John	Wick	53
	5	James	Bond	47
	6	Yuraj	Singh	43
*	NULL	NULL	NULL	NULL

Inserting data using column names,

The screenshot shows a SQL Enterprise Manager interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including 'mydbms', 'Tables', 'Columns', 'Indexes', 'Foreign Keys', 'Triggers', 'Views', 'Stored Procedures', 'Functions', 'sql_inventory', 'sql_invoicing', 'sql_store', and 'sys'. The 'persons' table is selected under 'Tables'. Below the tree, the 'Table: persons' information is shown, including columns: 'person_id' (int AI PK), 'firstname' (varchar(15)), and 'lastname' (varchar(25)).

The main query window displays the following SQL code:

```

27
28
29 • INSERT INTO persons (person_id,firstname, lastname, age) VALUES (1,'Sachin','Tendulkar',50);
30 • INSERT INTO persons (person_id,firstname, lastname, age) VALUES (2,'Taimur','Khan',10);
31 • INSERT INTO persons (person_id,firstname, lastname, age) VALUES (3,'Brad','Pitt',55);
32 • INSERT INTO persons (person_id,firstname, lastname, age) VALUES (4,'John','Wick',53);
33 • INSERT INTO persons (person_id,firstname, lastname, age) VALUES (5,'James','Bond',47);
34 • INSERT INTO persons (person_id,firstname, lastname, age) VALUES (6,'Yuraj','Singh',43);
35
36 • select * from persons;
37
38
39
40

```

The 'Result Grid' at the bottom shows the data returned by the query:

person_id	firstname	lastname	age
1	Sachin	Tendulkar	50
2	Taimur	Khan	10
3	Brad	Pitt	55
4	John	Wick	53
5	James	Bond	47
6	Yuraj	Singh	43
NULL	NULL	NULL	NULL

iii. 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. Select the following columns.

PAYMENT_ID from the payments table,

Amount from the payments table,

CLIENT_ID from the clients table,

Name as CLINET_NAME from the clients table,

Phone as CLIENT_PHONE from the clients table,

Name as PAYMENT_METHOD from PAYMENT_METHODS table.

The join query,

Navigator: SCHEMAS

Filter objects

- mydbms
 - Tables
 - persons
 - Views
 - Stored Procedures
 - Functions
 - sql_inventory
 - Tables
 - products
 - Views
 - Stored Procedures
 - Functions
 - sql_invoicing
 - Tables
 - clients
 - invoices
 - payment_methods
 - payments
 - Views
 - Stored Procedures

Administration Schemas

Information

Table: **payment_methods**

Columns:

payment_method_id tinyint AI PK
name varchar

Query 1 Assignment 1* SQL File 2* x

Limit to 1000 rows

```

53 • use sql_invoicing;
54
55 • select p.payment_id, p.amount, c.client_id, c.name as 'CLINET_NAME', c.phone as 'CLIENT_PHONE',
56     pm.name as 'PAYMENT_METHOD'
57 from payments as p inner join clients as c on p.client_id = c.client_id
58     inner join payment_methods as pm on pm.payment_method_id = p.payment_method
59
60
61
62
63
64
65

```

Result Grid

	payment_id	amount	client_id	CLINET_NAME	CLIENT_PHONE	PAYMENT_METHOD
1	1	8.18	5	Topidounge	971-888-9129	Credit Card
2	2	74.55	1	Vinte	315-252-7305	Credit Card
3	3	0.03	3	Yadel	415-144-6037	Credit Card
4	4	87.44	5	Topidounge	971-888-9129	Credit Card
5	5	80.31	3	Yadel	415-144-6037	Credit Card
6	6	68.10	3	Yadel	415-144-6037	Credit Card
7	7	32.77	5	Topidounge	971-888-9129	Credit Card
8	8	10.00	5	Topidounge	971-888-9129	Cash

Result 17 x Read Only

Another way of joining tables to get the same results,

Navigator: SCHEMAS

Filter objects

- mydbms
 - Tables
 - persons
 - Views
 - Stored Procedures
 - Functions
 - sql_inventory
 - Tables
 - products
 - Views
 - Stored Procedures
 - Functions
 - sql_invoicing
 - Tables
 - clients
 - invoices
 - payment_methods
 - payments
 - Views
 - Stored Procedures

Administration Schemas

Information

Table: **payment_methods**

Columns:

payment_method_id tinyint AI PK
name varchar

Query 1 Assignment 1* SQL File 2* x

Limit to 1000 rows

```

60
61
62
63 • select p.payment_id, p.amount, c.client_id, c.name as 'CLINET_NAME', c.phone as 'CLIENT_PHONE',
64     pm.name as 'PAYMENT_METHOD'
65 from
66     (payment_methods as pm inner join
67     (payments as p inner join clients as c on p.client_id = c.client_id)
68     on pm.payment_method_id = p.payment_method);
69
70
71
72
73

```

Result Grid

	payment_id	amount	client_id	CLINET_NAME	CLIENT_PHONE	PAYMENT_METHOD
1	1	8.18	5	Topidounge	971-888-9129	Credit Card
2	2	74.55	1	Vinte	315-252-7305	Credit Card
3	3	0.03	3	Yadel	415-144-6037	Credit Card
4	4	87.44	5	Topidounge	971-888-9129	Credit Card
5	5	80.31	3	Yadel	415-144-6037	Credit Card
6	6	68.10	3	Yadel	415-144-6037	Credit Card
7	7	32.77	5	Topidounge	971-888-9129	Credit Card
8	8	10.00	5	Topidounge	971-888-9129	Cash

- iv. Create a 'PAYMENT_DETAILS' table and insert records in it by executing the following join query.

Keeping an auto increment id field in the payment_details table.

'tbl_id' int(11) not null auto_increment

Here, the columns have been chosen with the same data types depending on the tables on which the aforementioned join query was mentioned.

The screenshot displays a database management interface with the following components:

- Navigator:** Shows a tree view of the database schema. The 'payment_details' table is selected under the 'sql_invoicing' database.
- Query Editor:** Contains the following SQL code:

```
73 • create table `payment_details` (  
74   `tbl_id` int(11) not null auto_increment,  
75   `payment_id` int(11) not null,  
76   `amount` decimal(9,2) not null,  
77   `client_id` int(11) not null,  
78   `client_name` varchar(50) not null,  
79   `client_phone` varchar(50) default null,  
80   `payment_method` varchar(50) NOT NULL,  
81   primary key (`tbl_id`));  
82  
83 • select * from payment_details;  
84  
85  
86
```
- Result Grid:** Shows the result of the 'select * from payment_details;' query. It displays a single row with all columns set to NULL.

tbl_id	payment_id	amount	client_id	client_name	client_phone	payment_method
NULL	NULL	NULL	NULL	NULL	NULL	NULL
- Table Information:** Provides details for the 'payment_details' table:
 - Columns:**
 - tbl_id: int AI PK
 - payment_id: int
 - amount: decimal(9,2)
 - client_id: int
 - client_name: varchar(50)
 - client_phone: varchar(50)
 - payment_method: varchar(50)

The below query shows the insertion of the outcome of the aforementioned join query into the recently created payment_details table.

Here, the join query's chosen columns have been properly mapped to the columns of the payment_details table.

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including tables like 'clients', 'invoices', 'payment_details', 'payment_methods', and 'payments'. The 'payment_details' table is selected, showing its columns: 'tbl_id', 'payment_id', 'amount', 'client_id', 'client_name', 'client_phone', and 'payment_method'.

The main pane displays a SQL query (Query 1) and its results. The query is as follows:

```

91
92 insert into payment_details (payment_id, amount, client_id, client_name, client_phone,
93 payment_method)
94 select p.payment_id, p.amount, c.client_id, c.name,
95 c.phone, pm.name
96 from
97 (payment_methods as pm inner join
98 (payments as p inner join clients as c on p.client_id = c.client_id)
99 on pm.payment_method_id = p.payment_method);
100
101
102 select * from payment_details;
103
104

```

The results are displayed in a table with the following columns: 'tbl_id', 'payment_id', 'amount', 'client_id', 'client_name', 'client_phone', and 'payment_method'. The data is as follows:

tbl_id	payment_id	amount	client_id	client_name	client_phone	payment_method
1	1	8.18	5	Topidounge	971-888-9129	Credit Card
2	2	74.55	1	Vinte	315-252-7305	Credit Card
3	3	0.03	3	Yadel	415-144-6037	Credit Card
4	4	87.44	5	Topidounge	971-888-9129	Credit Card
5	5	80.31	3	Yadel	415-144-6037	Credit Card
6	6	68.10	3	Yadel	415-144-6037	Credit Card
7	7	32.77	5	Topidounge	971-888-9129	Credit Card
8	8	10.00	5	Topidounge	971-888-9129	Cash
*	NULL	NULL	NULL	NULL	NULL	NULL

DML

- v. Refer 'PERSONS' table, modify the column constraint of AGE and LASTNAME and change it to not null.

The below syntax can be used to change NULL into NOT NULL constraint in MySQL.

ALTER TABLE table_name MODIFY column_name column_definition;

According to the snapshot below, the constraints for the "lastname" and "age" columns have been modified from NULL to NOT NULL.

It indicates that from now onwards, the user is unable to enter null values in these two columns.

The screenshot displays a database management interface with a left-hand 'Navigator' pane and a right-hand 'Query' editor.

Navigator Pane:

- SCHEMAS:** Filter objects. A tree view shows the database structure:
 - mydbms**
 - Tables**
 - persons**
 - Columns**: person_id, firstname, lastname, age
 - Indexes**
 - Foreign Keys**
 - Triggers**
 - Views**
 - Stored Procedures**
 - Functions**
 - sql_inventory**
 - Tables**: products
 - Views**
 - Stored Procedures**
 - Functions**
 - sql_invoicing**
 - Tables**: clients, invoices, payment_details, payment_methods, payments, persons
 - Views**
 - Stored Procedures**
 - Functions**
 - sql_store**
 - Tables**

Information Pane:

Table: persons

Columns:

Column Name	Data Type	Attributes
person_id	int	AI PK
firstname	varchar(15)	
lastname	varchar(25)	
age	int	

Query Editor:

The query editor shows a SQL script for modifying the 'persons' table constraints. The script includes comments and two ALTER TABLE statements.

```

125
126
127
128
129
130 /*Refer 'PERSONS' table, modify the column constraint of AGE and LASTNAME and
131    change it to not null*/
132
133
134 • alter table persons change `age` `age` int not null;
135 • alter table persons change `lastname` `lastname` varchar(25) not null;
136
137 /*after change the column constraints*/
138 /*below is the system generated sql script*/
139 • CREATE TABLE `persons` (
140     `person_id` int NOT NULL AUTO_INCREMENT,
141     `firstname` varchar(15) DEFAULT NULL,
142     `lastname` varchar(25) NOT NULL,
143     `age` int NOT NULL,
144     PRIMARY KEY (`person_id`)
145 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
146
147
148
149
150
151
152
153
154
155
156
157
158
159
  
```

vi. In the 'PERSONS' table, update the income of PERSON_ID 2, as 100 times of his age.

Adding a new column called "income," using decimal data type, since income is usually expressed in monetary terms.

The screenshot shows a database management interface with a Navigator pane on the left and a Query Editor on the right. The Navigator pane displays the 'mydbms' schema, including tables, indexes, foreign keys, triggers, views, stored procedures, functions, and sequences. The 'persons' table is selected, and its columns are listed: person_id, firstname, lastname, age, and income. The 'income' column is highlighted, and its definition is shown as 'income decimal(9,2)'. The Query Editor shows a SQL query to add the 'income' column to the 'persons' table and then select all data from the table.

```
/*In the 'PERSONS' table, update the income of PERSON_ID 2, as 100 times of his age*/  
  
alter table persons add column `income` decimal(9,2) default null;  
  
select * from persons;
```

person_id	firstname	lastname	age	income
1	Sachin	Tendulkar	50	NULL
2	Taimur	Khan	10	NULL
3	Brad	Pitt	55	NULL
4	John	Wick	53	NULL
5	James	Bond	47	NULL
6	Yuraj	Singh	43	NULL
*	NULL	NULL	NULL	NULL

Updating the 'income' column just for the person with ID 2 by multiplying the value of the 'age' column by 100.

Navigator

SCHEMAS

Filter objects

- mydbms
 - Tables
 - persons
 - Columns
 - person_id
 - firstname
 - lastname
 - age
 - income
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
 - sql_inventory
 - Tables
 - products
 - Views
 - Stored Procedures
 - Functions
 - sql_invoicing
 - Tables
 - clients
 - invoices
 - payment_details
 - payment_methods
 - payments
 - persons
 - Views
 - Stored Procedures
 - Functions
 - sql_store

Administration Schemas

Information

Column: income

Definition:
income decimal(9,2)

Query 1 Assignment 1* Assignment 1-2*

Limit to 1000 rows

```

157
158
159 • update persons set income = (age * 100) where person_id = 2;
160
161 • select * from persons;
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179

```

Result Grid Filter Rows: Edit: Export/Import:

	person_id	firstname	lastname	age	income
▶	1	Sachin	Tendulkar	50	NULL
	2	Taimur	Khan	10	1000.00
	3	Brad	Pitt	55	NULL
	4	John	Wick	53	NULL
	5	James	Bond	47	NULL
	6	Yuraj	Singh	43	NULL
*	NULL	NULL	NULL	NULL	NULL

- vii. Write a query to give 50 extra points to customers born before 1990. Use the customers table of SQL_STORE.

Identifying customers whose birth dates are less than January 1, 1990

Navigator

SCHEMAS

Filter objects

- mydbms
 - Tables
 - Views
 - Stored Procedures
 - Functions
- sql_inventory
- sql_invoicing
- sql_store
 - Tables
 - customers
 - order_item_notes
 - order_items
 - order_statuses
 - orders
 - products
 - shippers
 - Views
 - Stored Procedures
 - Functions
- sys

Query 1 Assignment 1* Assignment 1-2*

Limit to 1000 rows

```

164
165
166 /*Write a query to give 50 extra points to customers born before 1990. Use the customers table of SQL_STORE*/
167
168 • use sql_store;
169
170 • select * from customers;
171
172 /*find records where birthdate is less than 1990-01-01*/
173 • select * from customers where birth_date < '1990-01-01';
174
175
176
177
178
179
180
181
182
183
184
185
186

```

Result Grid

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	MA	2273
2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457
5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Administration Schemas

Information

Schema: sql_store

Updating the points column,

The screenshot displays a database management interface. On the left, a 'Navigator' pane shows a tree structure of schemas, with 'sql_store' selected. The main area shows a SQL editor with the following queries:

```

165
166 /*Write a query to give 50 extra points to customers born before 1990. Use the customers table of SQL_STORE*/
167
168 • use sql_store;
169
170 • select * from customers;
171
172 • select * from customers where birth_date < '1990-01-01';
173
174
175
176 • update customers set points = (points + 50) where birth_date < '1990-01-01';
177
178 • select * from customers;
179
180
181
182
183
184
185
186

```

Below the editor, a 'Result Grid' shows the data from the 'customers' table. The grid has columns: customer_id, first_name, last_name, birth_date, phone, address, city, state, and points. The data is as follows:

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	MA	2323
2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	997
3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	3017
4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	507
5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3725
6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1722
8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486
10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	846
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

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>