

# Team Project in Databases

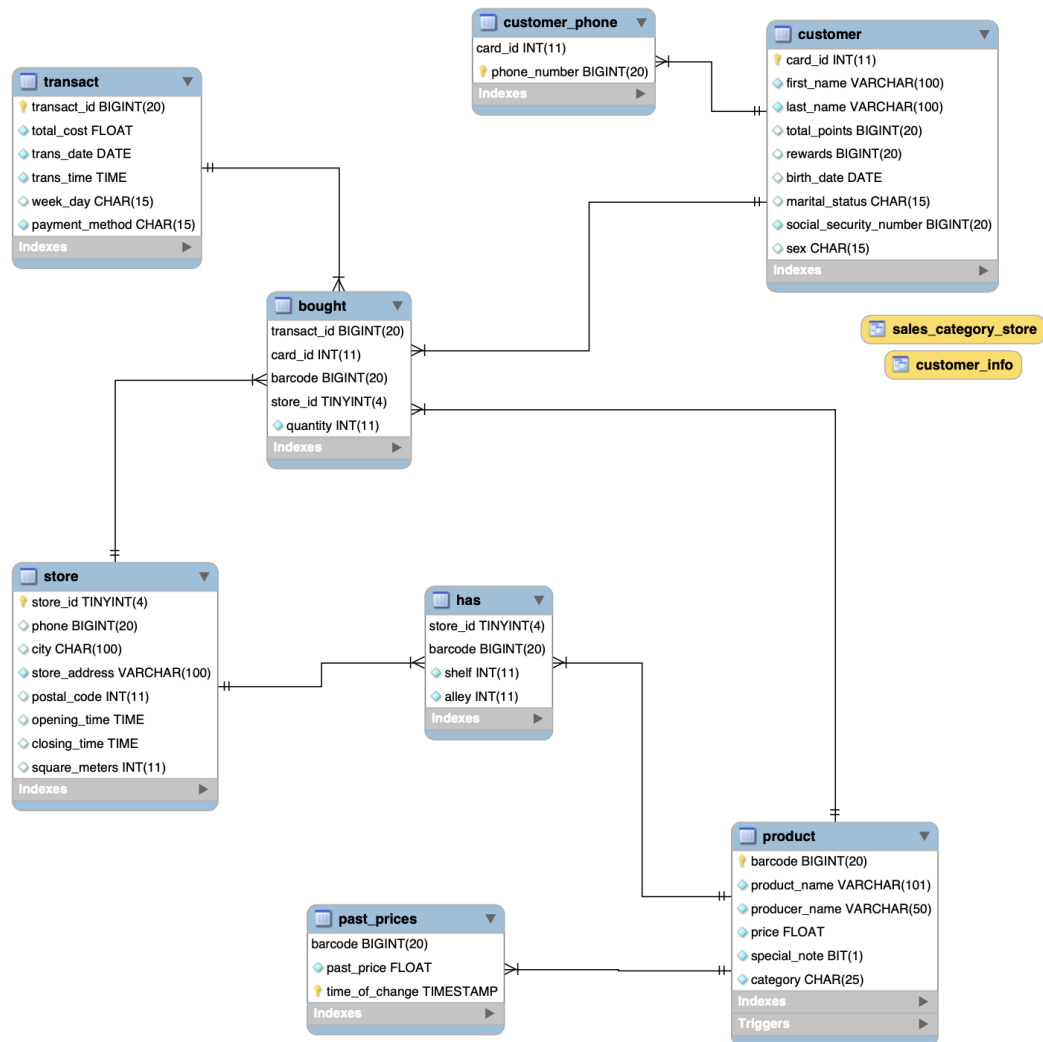
Alexandros Kyriakakis (03112163),  
Ioannis Alexopoulos (03117001),  
Christos Roumeliotis (03105218)

June 2020

# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Relational Diagram</b>              | <b>3</b>  |
| 1.1      | a. . . . .                             | 3         |
| 1.2      | b. . . . .                             | 4         |
|          | 1.2.1 Code to Create Indexes . . . . . | 4         |
|          | 1.2.2 Analysis . . . . .               | 4         |
| 1.3      | c. . . . .                             | 4         |
|          | 1.3.1 Requirements . . . . .           | 4         |
| 1.4      | d. . . . .                             | 5         |
|          | 1.4.1 Installation . . . . .           | 5         |
| <b>2</b> | <b>DDL</b>                             | <b>6</b>  |
| <b>3</b> | <b>SQL</b>                             | <b>8</b>  |
| 3.1      | Customer Data . . . . .                | 8         |
| 3.2      | Edit Data . . . . .                    | 10        |
| 3.3      | Indexes . . . . .                      | 10        |
| 3.4      | Product Data . . . . .                 | 10        |
| 3.5      | Search Per Condition . . . . .         | 18        |
| 3.6      | Add Stores . . . . .                   | 19        |
| 3.7      | Trigers . . . . .                      | 20        |
| 3.8      | Views . . . . .                        | 20        |
| <b>4</b> | <b>Installation Code</b>               | <b>21</b> |
| 4.1      | Add Data . . . . .                     | 21        |
| 4.2      | BackEnd Server . . . . .               | 23        |
| 4.3      | FrontEnd Server . . . . .              | 46        |
| <b>5</b> | <b>Video Indexing</b>                  | <b>47</b> |
| <b>6</b> | <b>Links</b>                           | <b>47</b> |

# 1 Relational Diagram



## 1.1 a.

According to our video at section 3, we used "NOT NULL" at an attribute only when the corresponding entity could not be initialized without this attributes. For example we could not add a customer without initializing its name. We chose everything respectively.

## 1.2 b.

### 1.2.1 Code to Create Indexes

```
1 use AlexJohnChris;
2 CREATE INDEX hour_trans ON transact(time);
3 CREATE INDEX special_bit ON product(special_note);
4 CREATE INDEX category_indx ON product(category);
```

### 1.2.2 Analysis

Considering that Mysql adds indexes on every primary key of our database, we chose to add the above three additional indexes on columns that are used frequently in our queries to speed up our database search time. For example the second index is clearly useful to the query presented below which makes extensive use of the special note column:

```
1 SELECT
2     A.category,
3     A.special_note,
4     COUNT(B.barcode) / COUNT(C.barcode) * 100 AS percentage
5 FROM
6     product A,
7     bought B,
8     product C
9 WHERE
10    A.barcode = B.barcode
11    AND B.barcode = C.barcode
12    AND A.special_note = 1
13 GROUP BY A.special_note , A.category
14 ORDER BY percentage DESC;
```

## 1.3 c.

The whole project's Database runs at an [AWS RDS](#) cloud server. Our Site is being hosted by [Heroku](#).

### 1.3.1 Requirements

- mysql 8.0.19
- flask 1.1.2
- mysql-connector 2.2.9
- numpy 1.17.4
- pandas 0.25.3

## 1.4 d.

### 1.4.1 Installation

1. At first, initialize a mysql database at either a [localhost](#) or a [server](#)
2. Then, run the following command in terminal, using your credentials in order to connect in mysql host:

```
1 $ mysql -h "server-name" -u "your_username" -p "your_password"
```

**Run the following inside mysql command prompt, strictly at this order,**

3. [AlexJohnChris.sql](#) to create the database.
4. [indexes.sql](#) to create the indexes.
5. [view1.sql](#) and [view2.sql](#) to create the views.
6. [past\\_price\\_trigger.sql](#) to create the trigger for auto-update past prices.
7. [addStores.sql](#) to add all the stores.

**Back in the terminal**

8. Run,

```
1 $ git clone https://github.com/AlexandrosKyriakakis/DataBase.git
2 $ cd DataBase
3 $ git clone https://github.com/AlexandrosKyriakakis/MarketDataset.git
```

9. Add your database credentials at the top of each of the following files,

- [addCustomersAndPhone.py](#)
- [addProductsPastPricesHas.py](#)
- [addTransactionsBought.py](#)
- [server\\_guest.py](#)

10. Run the following strictly at this order,

```
1 $ pip3 install -r requirements.txt
2 $ python3 ./addData/addCustomersAndPhone.py
3 $ python3 ./addData/addProductsPastPricesHas.py
4 $ python3 ./addData/addTransactionsBought.py
```

11. Now, that the database is full with random generated data, start the back-end server to finish the installation,

```
1 $ python3 server_guest.py
```

12. Open your favorite browser and type <http://localhost:8587> to preview the website.

## 2 DDL

```
1 CREATE DATABASE `AlexJohnChris`;
2 ALTER DATABASE AlexJohnChris CHARACTER
3 SET utf8
4 COLLATE utf8_bin;
5 USE `AlexJohnChris`;
6 /* create table Store*/
7 CREATE TABLE store
8 (
9     store_id TINYINT NOT NULL
10    AUTO_INCREMENT,
11     phone BIGINT,
12     city CHAR
13    (100),
14     store_address VARCHAR
15    (100) NOT NULL,
16     postal_code INT,
17     opening_time TIME,
18     closing_time TIME,
19     square_meters INT,
20     CONSTRAINT PKstore PRIMARY KEY
21    (store_id));
22
23 /* create table Customer*/
24 CREATE TABLE customer
25 (
26     card_id INT NOT NULL
27    AUTO_INCREMENT,
28     first_name VARCHAR
29    (100) NOT NULL,
30     last_name VARCHAR
31    (100) NOT NULL,
32     total_points BIGINT DEFAULT 0,
33     rewards BIGINT DEFAULT 0,
34     birth_date DATE,
35     marital_status CHAR
36    (15),
37     social_security_number BIGINT NOT NULL,
38     sex CHAR
39    (15),
40     CONSTRAINT PKcustomer PRIMARY KEY
41    (card_id));
42
43 /*customer_phone multivalued attribute of customer*/
44 CREATE TABLE customer_phone
45 (
46     card_id INT NOT NULL,
```

```

47     phone_number BIGINT NOT NULL,
48     CONSTRAINT PKcustomer_phone PRIMARY KEY(card_id, phone_number),
49     CONSTRAINT FKcustomer_phone FOREIGN KEY (card_id) REFERENCES customer(card_id)
    ↪ ON UPDATE CASCADE ON DELETE CASCADE
50 );
51
52 /* create table Transaction*/
53 CREATE TABLE transact
54 (
55     transact_id BIGINT NOT NULL
56     AUTO_INCREMENT,
57     total_cost FLOAT NOT NULL,
58     trans_date DATE NOT NULL,
59     trans_time TIME NOT NULL,
60     week_day CHAR
61     (15),
62     payment_method CHAR
63     (15) NOT NULL,
64     CONSTRAINT PKtransact PRIMARY KEY
65     (transact_id));
66
67 /* create table Product*/
68 CREATE TABLE product
69 (
70     barcode BIGINT NOT NULL,
71     product_name VARCHAR(101) NOT NULL,
72     /* Added name*/
73     producer_name VARCHAR(50) NOT NULL,
74     /* Added producer name*/
75     price FLOAT NOT NULL,
76     special_note BIT(1) NOT NULL,
77     /* To kana bit gt mas niazei mono an anoikei h den anhkei sto katasthma 0 ->
    ↪ gia kseno 1 gia AB*/
78     category CHAR(25) NOT NULL,
79     CONSTRAINT PKvehicle PRIMARY KEY(barcode)
80 );
81
82 /* works : 1-n relationship (optional,optional)*/
83 CREATE TABLE has
84 (
85     store_id TINYINT NOT NULL,
86     barcode BIGINT NOT NULL,
87     shelf INT NOT NULL,
88     alley INT NOT NULL,
89     CONSTRAINT PKhas PRIMARY KEY (store_id, barcode),
90     CONSTRAINT FKistore FOREIGN KEY (store_id) REFERENCES store(store_id) ON
    ↪ UPDATE CASCADE ON DELETE CASCADE,

```

```

91     CONSTRAINT FK2has_products FOREIGN KEY (barcode) REFERENCES product(barcode)
    ↪ ON UPDATE CASCADE ON DELETE CASCADE
92 );
93
94 /* past_prices 1-n relationship(optional,mandatory)*/
95 CREATE TABLE past_prices
96 (
97     barcode BIGINT NOT NULL,
98     past_price FLOAT NOT NULL,
99     time_of_change TIMESTAMP NOT NULL,
100     CONSTRAINT PKpast_prices PRIMARY KEY (barcode,time_of_change),/*SOS*/
101     CONSTRAINT FKhad_price FOREIGN KEY (barcode) REFERENCES product(barcode) ON
    ↪ UPDATE CASCADE ON DELETE CASCADE
102 );
103
104
105 /* rents: n-m-k-l relationship(optional,optional,optional,mandatory) the
    ↪ transaction part is mandatory*/
106 /* payment transactions*/
107 CREATE TABLE bought
108 (
109     transact_id BIGINT NOT NULL,
110     card_id INT NOT NULL,
111     barcode BIGINT NOT NULL,
112     store_id TINYINT NOT NULL,
113     quantity INT NOT NULL,
114     CONSTRAINT PKbought PRIMARY KEY (transact_id, card_id, barcode, store_id),
115     CONSTRAINT FK1product FOREIGN KEY (barcode) REFERENCES product(barcode) ON
    ↪ UPDATE CASCADE ON DELETE CASCADE,
116     CONSTRAINT FK2customer FOREIGN KEY (card_id) REFERENCES customer(card_id) ON
    ↪ UPDATE CASCADE ON DELETE CASCADE,
117     CONSTRAINT FK3transact FOREIGN KEY (transact_id) REFERENCES
    ↪ transact(transact_id) ON UPDATE CASCADE ON DELETE CASCADE,
118     CONSTRAINT FK4store FOREIGN KEY (store_id) REFERENCES store(store_id) ON
    ↪ UPDATE CASCADE ON DELETE CASCADE
119 );
120

```

## 3 SQL

### 3.1 Customer Data

```

1  select hour(trans_time), count(hour(trans_time))
2  from transact
3  where transact_id
4     in (
5         select transact_id
6     from bought

```



```

7  where card_id = 99
8      )
9  group by hour(trans_time)
10 order by hour(trans_time)

1  SELECT
2      distinct b.card_id, b.store_id
3  FROM
4      bought as b, store as s
5  where
6      b.store_id = s.store_id
7      AND b.card_id = 99
8  order by card_id, store_id
9
10 SELECT card_id, barcode, sum(quantity)
11 FROM bought
12 where card_id = 99
13 group by barcode
14 order by sum(quantity) desc
15 limit 10
16
17
18 SELECT
19     b
20     .card_id, MONTHNAME
21     (t.trans_date),
22     cast
23     (AVG
24     (t.total_cost) as decimal
25     (6,2))
26 FROM
27     transact AS t,
28     bought AS b
29 WHERE
30     b.card_id = 99
31     AND b.transact_id = t.transact_id
32 GROUP BY MONTHNAME
33     (t.trans_date)
34 ORDER BY MONTH
35     (t.trans_date);
36
37
38 SELECT
39     b.card_id,
40     WEEK(t.trans_date) Week,
41     cast(AVG(total_cost) as decimal(6,2))
42 FROM
43     bought AS b,

```

```

44     transact AS t
45 WHERE
46     b.card_id = 99
47     AND b.transact_id = t.transact_id
48 GROUP BY WEEK(t.trans_date)

```

### 3.2 Edit Data

```

1  insert into store
2      (phone,city,store_address,postal_code,
3       opening_time,closing_time ,square_meters)
4  values
5      ('1111111', 'Test', 'Test', '1111',
6       '08:00:00', '21:00:00', '1111')

1  DELETE FROM store WHERE store_id = 11

1  update store
2  set
3      phone = '222222' , city = 'NewTest' ,
4      store_address = 'NewTest' , postal_code = '22222',
5      opening_time = '09:00:00' , closing_time = '22:00:00' ,
6      square_meters = '22222' where store_id = 11;

```

We follow the same pattern for editing both Products and Customers.

### 3.3 Indexes

```

1  use AlexJohnChris;
2  CREATE INDEX hour_trans ON transact(time);
3  CREATE INDEX special_bit ON product(special_note);
4  CREATE INDEX category_indx ON product(category);

```

### 3.4 Product Data

```

1  SELECT distinct
2      cast(SUM(
3  IF(A.age < 20, 1, 0))/totals.total _ 100 as decimal
4  (20,2)) AS 'Under 20',
5      cast
6  (SUM
7  (
8  IF(A.age BETWEEN 20 AND 29, 1, 0))/totals.total _ 100 as decimal
9  (20,2)) AS '20 - 29',
10     cast
11  (SUM
12  (
13  IF(A.age BETWEEN 30 AND 39, 1, 0))/totals.total _ 100 as decimal
14  (20,2)) AS '30 - 39',

```

```

15     cast
16 (SUM
17 (
18 IF(A.age BETWEEN 40 AND 49, 1, 0))/totals.total _ 100 as decimal
19 (20,2)) AS '40 - 49',
20     cast
21 (SUM
22 (
23 IF(A.age BETWEEN 50 AND 59, 1, 0))/totals.total _ 100 as decimal
24 (20,2)) AS '50 - 59',
25     cast
26 (SUM
27 (
28 IF(A.age BETWEEN 60 AND 69, 1, 0))/totals.total _ 100 as decimal
29 (20,2)) AS '60 - 69',
30     cast
31 (SUM
32 (
33 IF(A.age >= 70, 1, 0))/totals.total _ 100 as decimal
34 (20,2)) AS 'Over 70',
35     HOUR
36 (C.trans_time) AS Hour
37 FROM
38 (SELECT distinct
39     YEAR(CURDATE()) - YEAR(birth_date) -
40 IF(STR_TO_DATE(CONCAT(YEAR(CURDATE()),
41     '-', MONTH(birth_date), '-', DAY(birth_date)), '%Y-%c-%e') > CURDATE(),
42     1, 0) AS age,
43     card_id
44 FROM
45     customer) A,
46 (SELECT _
47 from bought
48 group by transact_id)
49 B
50 LEFT JOIN
51     transact C on B.transact_id = C.transact_id,
52 (SELECT distinct
53     COUNT(D.transact_id) as total, HOUR(D.trans_time) as total_hour
54 FROM
55     transact D
56 GROUP BY HOUR(D.trans_time)
57 ORDER BY HOUR(D.trans_time)
58 ) totals
59 WHERE
60     A.card_id = B.card_id
61     AND B.transact_id = C.transact_id
62     AND totals.total_hour = HOUR

```

```

62 (C.trans_time)
63 GROUP BY HOUR
64 (C.trans_time)
65 ORDER BY HOUR
66 (C.trans_time)

1  (select p.category, h.store*id, COUNT(*) as counter
2  FROM product p, has h, bought b
3  WHERE b.store*id = h.store_id
4  AND h.barcode = b.barcode
5  AND h.barcode = p.barcode
6  AND p.category = 'Φρέσκα Προϊόντα'
7  GROUP BY h.store_id
8  ORDER BY counter desc
9  limit 1)
10 UNION
11 (select p.category, h.store_id, COUNT(*) as counter
12 FROM product p, has h, bought b
13 WHERE b.store*id = h.store_id
14 AND h.barcode = b.barcode
15 AND h.barcode = p.barcode
16 AND p.category = 'Ποϊόντα Ψυγείου'
17 GROUP BY h.store_id
18 ORDER BY counter desc
19 limit 1)
20 UNION
21 (select p.category, h.store_id, COUNT(*) as counter
22 FROM product p, has h, bought b
23 WHERE b.store*id = h.store_id
24 AND h.barcode = b.barcode
25 AND h.barcode = p.barcode
26 AND p.category = 'Κατοικίδια'
27 GROUP BY h.store_id
28 ORDER BY counter desc
29 limit 1)
30 UNION
31 (select p.category, h.store_id, COUNT(*) as counter
32 FROM product p, has h, bought b
33 WHERE b.store*id = h.store_id
34 AND h.barcode = b.barcode
35 AND h.barcode = p.barcode
36 AND p.category = 'Κ&β&α'
37 GROUP BY h.store_id
38 ORDER BY counter desc
39 limit 1)
40 UNION
41 (select p.category, h.store_id, COUNT(*) as counter
42 FROM product p, has h, bought b

```

```

43 WHERE b.store*id = h.store_id
44 AND h.barcode = b.barcode
45 AND h.barcode = p.barcode
46 AND p.category = 'Είδη Σπιτιού'
47 GROUP BY h.store_id
48 ORDER BY counter desc
49 limit 1)
50 UNION
51 (select p.category, h.store_id, COUNT(*) as counter
52 FROM product p, has h, bought b
53 WHERE b.store_id = h.store_id
54 AND h.barcode = b.barcode
55 AND h.barcode = p.barcode
56 AND p.category = 'Προσωπικής Περιποίησης'
57 GROUP BY h.store_id
58 ORDER BY counter desc
59 limit 1)
60 UNION
61 (select p.category, h.store_id, COUNT(\N*) as counter
62 FROM product p, has h, bought b
63 WHERE b.store_id = h.store_id
64 AND h.barcode = b.barcode
65 AND h.barcode = p.barcode
66 AND p.category = 'Είδη Σπιτιού'
67 GROUP BY h.store_id
68 ORDER BY counter desc
69 limit 1)

1 (SELECT
2 A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
3 FROM
4 bought A,
5 bought B,
6 has C,
7 has D
8 WHERE
9 A.transact_id = B.transact_id
10 AND A.barcode > B.barcode
11 AND A.barcode = C.barcode
12 AND A.store_id = C.store_id
13 AND B.barcode = D.barcode
14 AND B.store_id = 1
15 AND B.store_id = D.store_id
16 GROUP BY A.barcode , B.barcode, A.store_id
17 ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
18 LIMIT 1)
19 union
20 (SELECT

```

```

21  A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
22  FROM
23  bought A,
24  bought B,
25  has C,
26  has D
27  WHERE
28  A.transact_id = B.transact_id
29  AND A.barcode > B.barcode
30  AND A.barcode = C.barcode
31  AND A.store_id = C.store_id
32  AND B.barcode = D.barcode
33  AND B.store_id = 2
34  AND B.store_id = D.store_id
35  GROUP BY A.barcode , B.barcode, A.store_id
36  ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
37  LIMIT 1)
38  union
39  (SELECT
40  A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
41  FROM
42  bought A,
43  bought B,
44  has C,
45  has D
46  WHERE
47  A.transact_id = B.transact_id
48  AND A.barcode > B.barcode
49  AND A.barcode = C.barcode
50  AND A.store_id = C.store_id
51  AND B.barcode = D.barcode
52  AND B.store_id = 3
53  AND B.store_id = D.store_id
54  GROUP BY A.barcode , B.barcode, A.store_id
55  ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
56  LIMIT 1)
57  union
58  (SELECT
59  A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
60  FROM
61  bought A,
62  bought B,
63  has C,
64  has D
65  WHERE
66  A.transact_id = B.transact_id
67  AND A.barcode > B.barcode
68  AND A.barcode = C.barcode

```

```

69  AND A.store_id = C.store_id
70  AND B.barcode = D.barcode
71  AND B.store_id = 4
72  AND B.store_id = D.store_id
73  GROUP BY A.barcode , B.barcode, A.store_id
74  ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
75  LIMIT 1)
76  union
77  (SELECT
78  A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
79  FROM
80  bought A,
81  bought B,
82  has C,
83  has D
84  WHERE
85  A.transact_id = B.transact_id
86  AND A.barcode > B.barcode
87  AND A.barcode = C.barcode
88  AND A.store_id = C.store_id
89  AND B.barcode = D.barcode
90  AND B.store_id = 5
91  AND B.store_id = D.store_id
92  GROUP BY A.barcode , B.barcode, A.store_id
93  ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
94  LIMIT 1)
95  union
96  (SELECT
97  A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
98  FROM
99  bought A,
100  bought B,
101  has C,
102  has D
103  WHERE
104  A.transact_id = B.transact_id
105  AND A.barcode > B.barcode
106  AND A.barcode = C.barcode
107  AND A.store_id = C.store_id
108  AND B.barcode = D.barcode
109  AND B.store_id = 6
110  AND B.store_id = D.store_id
111  GROUP BY A.barcode , B.barcode, A.store_id
112  ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
113  LIMIT 1)
114  union
115  (SELECT
116  A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id

```

```

117 FROM
118 bought A,
119 bought B,
120 has C,
121 has D
122 WHERE
123 A.transact_id = B.transact_id
124 AND A.barcode > B.barcode
125 AND A.barcode = C.barcode
126 AND A.store_id = C.store_id
127 AND B.barcode = D.barcode
128 AND B.store_id = 7
129 AND B.store_id = D.store_id
130 GROUP BY A.barcode , B.barcode, A.store_id
131 ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
132 LIMIT 1)
133 union
134 (SELECT
135 A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
136 FROM
137 bought A,
138 bought B,
139 has C,
140 has D
141 WHERE
142 A.transact_id = B.transact_id
143 AND A.barcode > B.barcode
144 AND A.barcode = C.barcode
145 AND A.store_id = C.store_id
146 AND B.barcode = D.barcode
147 AND B.store_id = 8
148 AND B.store_id = D.store_id
149 GROUP BY A.barcode , B.barcode, A.store_id
150 ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
151 LIMIT 1)
152 union
153 (SELECT
154 A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
155 FROM
156 bought A,
157 bought B,
158 has C,
159 has D
160 WHERE
161 A.transact_id = B.transact_id
162 AND A.barcode > B.barcode
163 AND A.barcode = C.barcode
164 AND A.store_id = C.store_id

```



```

165 AND B.barcode = D.barcode
166 AND B.store_id = 9
167 AND B.store_id = D.store_id
168 GROUP BY A.barcode , B.barcode, A.store_id
169 ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
170 LIMIT 1)
171 union
172 (SELECT
173 A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
174 FROM
175 bought A,
176 bought B,
177 has C,
178 has D
179 WHERE
180 A.transact_id = B.transact_id
181 AND A.barcode > B.barcode
182 AND A.barcode = C.barcode
183 AND A.store_id = C.store_id
184 AND B.barcode = D.barcode
185 AND B.store_id = 10
186 AND B.store_id = D.store_id
187 GROUP BY A.barcode , B.barcode, A.store_id
188 ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
189 LIMIT 1)

1 SELECT
2 A.barcode, B.barcode, COUNT(A.barcode)
3 FROM
4 bought A,
5 bought B
6 WHERE
7 A.transact_id = B.transact_id
8 AND A.barcode > B.barcode
9 GROUP BY A.barcode , B.barcode
10 ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
11 LIMIT 100;

1 SELECT
2 A.store_id, A.shelf, A.alley, COUNT(B.barcode)
3 FROM
4 has A,
5 bought B
6 WHERE
7 A.barcode = B.barcode
8 GROUP BY B.barcode , A.store_id , A.shelf , A.alley
9 ORDER BY COUNT(B.barcode) DESC
10 LIMIT 100;

```

```

1  SELECT
2      HOUR(trans_time),
3      CAST(SUM(A.total_cost) AS DECIMAL (20 , 2 ))
4  FROM
5      transact A,
6      bought B
7  WHERE
8      A.transact_id = B.transact_id
9  GROUP BY HOUR(trans_time)
10 ORDER BY SUM(A.total_cost) DESC , HOUR(trans_time)

1  SELECT
2      A.category,
3      A.special_note,
4      COUNT(B.barcode) / COUNT(C.barcode) * 100 AS percentage
5  FROM
6      product A,
7      bought B,
8      product C
9  WHERE
10     A.barcode = B.barcode
11     AND B.barcode = C.barcode
12     AND A.special_note = 1
13 GROUP BY A.special_note , A.category
14 ORDER BY percentage DESC;

```

### 3.5 Search Per Condition

```

1  use AlexJohnChris;
2  select
3      transact.transact_id, transact.total_cost,
4      transact.trans_date, transact.trans_time,
5      transact.week_day, transact.payment_method,
6      bought.card_id, bought.barcode, bought.quantity
7  from transact, bought
8  where
9      bought.transact_id = transact.transact_id
10     and bought.store_id = 2
11     and month(transact.trans_date) = month('1980-11-01')
12     and year(transact.trans_date) = year('1980-11-01')
13     and bought.quantity >= 4
14     and bought.barcode = 5000665
15     and transact.total_cost >= 3.0
16     and transact.payment_method = 'Card'
17     and bought.barcode in (select barcode
18                             from product
19                             where category = 'Είδη Σπιτιού')
20 limit 1000;

```

### 3.6 Add Stores

```
1  USE `AlexJohnChris`;
2  INSERT INTO store
   ↪ (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
3    VALUES (2108476357,'Αθήνα','Κουμουνδούρου & Παλαιολόγου
   ↪ 46-48',17937,'08:00:00','21:00:00',240);
4
5  INSERT INTO store
   ↪ (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
6    VALUES (2108476357,'Καρπενήσι','Αγ.
   ↪ Γεώργιος',36071,'08:00:00','21:00:00',292);
7
8  INSERT INTO store
   ↪ (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
9    VALUES (2108476357,'Αθήνα','Θηβών-Ιδομενέως & Αωνυμίου
   ↪ 0δού',13121,'08:00:00','21:00:00',372);
10
11 INSERT INTO store
   ↪ (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
12    VALUES (2108476357,'Αθήνα','Λ. Ηλιουπόλεως 4 & Κασσιόπης 3
   ↪ ',17237,'08:00:00','21:00:00',534);
13
14 INSERT INTO store
   ↪ (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
15    VALUES (2108476357,'Αθήνα','Φιλαδελφείας
   ↪ 101',13671,'08:00:00','21:00:00',4234);
16
17 INSERT INTO store
   ↪ (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
18    VALUES (2108476357,'Αθήνα','Πεύκων
   ↪ 26',15126,'08:00:00','21:00:00',423);
19
20 INSERT INTO store
   ↪ (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
21    VALUES (2108476357,'Χαλκίδα','NEA
   ↪ APTAKH',34600,'08:00:00','21:00:00',123);
22
23 INSERT INTO store
   ↪ (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
24    VALUES (2108476357,'Χαλκίδα','Γ. Παπανδρέου & Άνδρου
   ↪ 1',34100,'08:00:00','21:00:00',45);
25
26 INSERT INTO store
   ↪ (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
27    VALUES (2108476357,'Αθήνα','Μακεδονίας 2 & Τενέδου
   ↪ 8',15235,'08:00:00','21:00:00',3124);
28
```

```

29 INSERT INTO store
   ↪ (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
30 VALUES (2108476357,'Αθήνα','Λ. Λαυρίου 160 & Χρυσανθέμων
   ↪ ',15354,'08:00:00','21:00:00',23);

```

### 3.7 Triggers

```

1 use AlexJohnChris;
2 DELIMITER //
3 DROP TRIGGER IF EXISTS past_p
4 //
5 CREATE TRIGGER past_p AFTER
6 UPDATE
7 ON product
8 FOR EACH ROW
9 BEGIN
10     IF (not NEW.price <=> OLD.price) THEN
11         INSERT INTO past_prices
12             (barcode,past_price,time_of_change)
13         VALUES
14             (OLD.barcode, OLD.price, NOW());
15     END
16 IF;
17 END;//
18 DELIMITER ;

```

### 3.8 Views

```

1 use AlexJohnChris;
2 CREATE VIEW sales_category_store
3 AS
4 SELECT B.transact_id, B.card_id, H.barcode, B.quantity, P.category, H.store_id
5 FROM
6     bought B, has H, product P
7 WHERE B.barcode = H.barcode
8 AND H.barcode = P.barcode
9 ORDER BY P.category, H.store_id

1 use AlexJohnChris;
2 CREATE VIEW customer_info
3 AS
4 SELECT DISTINCT c.card_id, c.first_name, c.last_name, c.total_points, c.rewards,
   ↪ c.birth_date, c.marital_status, c.social_security_number, c.sex,
   ↪ p.phone_number
5 FROM
6     customer c
7 LEFT JOIN
8     customer_phone p
9 on c.card_id = p.card_id

```

## 4 Installation Code

### 4.1 Add Data

```
1 import mysql.connector
2 import pandas as pd
3 import numpy as np
4 People = pd.read_csv("./MarketDataset/data/RandomPeople.csv")
5 mydb = mysql.connector.connect(
6     host = "****",
7     user = "****",
8     passwd = "****",
9     database = "AlexJohnChris"
10 )
11
12 mycursor = mydb.cursor()
13 rewardAll = np.arange(1,100,1)
14 pointsAll = np.arange(1,1000,1)
15 phones = np.arange(2100000000,2109999999,10)
16 for i in range(len(People)):
17     # ssn,first_name,last_name,email,gender,birth
18     first_name = People['first_name'][i].replace("'", "").replace('"', "")
19     last_name = People['last_name'][i].replace("'", "").replace('"', "")
20     email = People['email'][i]
21     sex = People['gender'][i]
22     total_points = np.random.choice(pointsAll)
23     rewards = np.random.choice(rewardAll)
24     mar = 'Widow' if sex == 'Female' else 'Widower'
25     social_security_number = People['ssn'][i].replace('-', '')
26     birth_date = People['birth'][i]
27     marital_status =
28     ↪ np.random.choice(['Married', 'Single', 'Divorced', 'Engaged', mar])
29     sqlFormula = """INSERT INTO customer
30     ↪ (first_name,last_name,birth_date,marital_status,sex,social_security_number,total_points,re
31     ↪
32     VALUES
33     ↪ ('{}','{}','{}','{}','{}',{},{},{})""".format(first_name,last_name,birth_date,marital_status,s
34     mycursor.execute(sqlFormula)
35     mydb.commit() #Save Data
36     for k in range(3):
37         sqlFormula = """INSERT INTO customer_phone (card_id,phone_number)
38         VALUES ({},{})""".format(i+1,np.random.choice(phones))
39         mycursor.execute(sqlFormula)
40     mydb.commit() #Save Data
41
42 import mysql.connector
43 import pandas as pd
44 import numpy as np
45 People = pd.read_csv("./MarketDataset/data/RandomPeople.csv")
```

```

5 mydb = mysql.connector.connect(
6     host = "****",
7     user = "****",
8     passwd = "****",
9     database = "AlexJohnChris"
10 )
11
12 mycursor = mydb.cursor()
13 rewardAll = np.arange(1,100,1)
14 pointsAll = np.arange(1,1000,1)
15 phones = np.arange(2100000000,2109999999,10)
16 for i in range(len(People)):
17     # ssn,first_name,last_name,email,gender,birth
18     first_name = People['first_name'][i].replace("'", "").replace('"', "")
19     last_name = People['last_name'][i].replace("'", "").replace('"', "")
20     email = People['email'][i]
21     sex = People['gender'][i]
22     total_points = np.random.choice(pointsAll)
23     rewards = np.random.choice(rewardAll)
24     mar = 'Widow' if sex == 'Female' else 'Widower'
25     social_security_number = People['ssn'][i].replace('-', '')
26     birth_date = People['birth'][i]
27     marital_status =
28         ↪ np.random.choice(['Married', 'Single', 'Divorced', 'Engaged', mar])
29     sqlFormula = """INSERT INTO customer
30         ↪ (first_name,last_name,birth_date,marital_status,sex,social_security_number,total_points,re
31         ↪
32         VALUES
33         ↪ ('{}','{}','{}','{}','{}',{},{},{})""".format(first_name,last_name,birth_date,marital_status,s
34     mycursor.execute(sqlFormula)
35     mydb.commit() #Save Data
36     for k in range(3):
37         sqlFormula = """INSERT INTO customer_phone (card_id,phone_number)
38             VALUES ({},{})""".format(i+1,np.random.choice(phones))
39         mycursor.execute(sqlFormula)
40     mydb.commit() #Save Data
41
42 import mysql.connector
43 import random as rd
44 import pandas as pd
45 import numpy as np
46 ab = pd.read_csv("./MarketDataset/data/AB.csv")
47 mydb = mysql.connector.connect(
48     host = "****",
49     user = "****",
50     passwd = "****",
51     database = "AlexJohnChris"
52 )

```

```

12
13 mycursor = mydb.cursor()
14 cardAll = np.arange(1,1000,1)
15 storeAll = np.arange(1,11,1)
16 CostAll = np.arange(1,200,0.1)
17 year,month,day = np.arange(1970,2019,1), np.arange(1,12,1), np.arange(1,29,1)
18 hour,minute,second = np.arange(10,23,1), np.arange(10,59,1), np.arange(10,59,1)
19 for i in range(5000):
20     total_cost = np.random.choice(CostAll)
21     trans_date =
22     ↪     "-".join([str(np.random.choice(year)),str(np.random.choice(month)),
23     ↪     str(np.random.choice(day))])
24     trans_time =
25     ↪     "-".join([str(np.random.choice(hour)),str(np.random.choice(minute)),
26     ↪     str(np.random.choice(second))])
27     payment_method = np.random.choice(['Cash','Card'])
28     week_day =
29     ↪     np.random.choice(['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'])
30     sqlFormula = """INSERT INTO transact
31     ↪     (total_cost,trans_date,trans_time,week_day,payment_method)
32     VALUES
33     ↪     ({},{},{},{},{})""".format(total_cost,trans_date,trans_time,week_day,payment_method)
34     mycursor.execute(sqlFormula)
35     mydb.commit() #Save Data
36     card_id = np.random.choice(cardAll)
37     store_id = np.random.choice(storeAll)
38     randomMy = np.random.choice(range(4500))
39     for j in range(np.random.choice(range(100))):
40         barcode = ab['barcode'][j + randomMy + 1]
41         quantity = np.random.choice(range(1,6))
42         sqlFormula = """INSERT INTO bought
43         ↪     (transact_id,card_id,store_id,quantity,barcode)
44         VALUES
45         ↪     ({},{},{},{},{})""".format(i+1,card_id,store_id,quantity,barcode)
46         mycursor.execute(sqlFormula)
47     mydb.commit()

```

## 4.2 BackEnd Server

```

1 import os
2 from flask import Flask, render_template, request
3 from flask_mysql import MySQL
4
5 app = Flask(__name__)
6 app.config["MYSQL_USER"] = "*****"
7 app.config["MYSQL_PASSWORD"] = "*****"
8 app.config["MYSQL_HOST"] = "*****"
9 app.config["MYSQL_DB"] = "AlexJohnChris"

```

```

10 mysql = MySQL(app)
11
12
13 def shutdown_server():
14     func = request.environ.get("werkzeug.server.shutdown")
15     if func is None:
16         raise RuntimeError("Not running with the Werkzeug Server")
17     func()
18
19
20 @app.route("/")
21 def mainIndex():
22     return render_template("layout.html")
23
24
25 @app.route("/product_bought_together", methods=["GET", "POST"])
26 def product_bought_together():
27     cur = mysql.connection.cursor()
28     my_query = """SELECT
29         A.barcode, B.barcode, COUNT(A.barcode)
30     FROM
31         bought A,
32         bought B
33     WHERE
34         A.transact_id = B.transact_id
35         AND A.barcode > B.barcode
36     GROUP BY A.barcode , B.barcode
37     ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
38     LIMIT 100;
39     """
40     print(my_query)
41     cur.execute(my_query)
42     results = cur.fetchall()
43     return render_template("music3.html", results=results)
44
45
46 @app.route("/popular_positions", methods=["GET", "POST"])
47 def popular_positions():
48     cur = mysql.connection.cursor()
49     my_query = """SELECT
50         A.store_id, A.shelf, A.alley, COUNT(B.barcode)
51     FROM
52         has A,
53         bought B
54     WHERE
55         A.barcode = B.barcode
56     GROUP BY B.barcode , A.store_id , A.shelf , A.alley
57     ORDER BY COUNT(B.barcode) DESC

```



```

58 LIMIT 100;
59 """
60     print(my_query)
61     cur.execute(my_query)
62     results = cur.fetchall()
63     return render_template("music4.html", results=results)
64
65
66 @app.route("/label_trust", methods=["GET", "POST"])
67 def label_trust():
68     cur = mysql.connection.cursor()
69     my_query = """SELECT
70     A.category,
71     COUNT(B.barcode) / E.total * 100 AS percentage
72 FROM
73     product A,
74     bought B,
75     (SELECT COUNT(C.barcode) as total, D.category as categor
76 FROM bought C, product D
77 WHERE C.barcode = D.barcode
78 GROUP BY D.category)E
79 WHERE
80     A.barcode = B.barcode
81     and A.category = E.categor
82     AND A.special_note = 1
83 GROUP BY A.special_note , A.category
84 ORDER BY percentage DESC;
85 """
86     print(my_query)
87     cur.execute(my_query)
88     results = cur.fetchall()
89     return render_template("music5.html", results=results)
90
91
92 @app.route("/hours_spent_more", methods=["GET", "POST"])
93 def hours_spent_more():
94     cur = mysql.connection.cursor()
95     my_query = """SELECT
96     HOUR(trans_time),
97     CAST(SUM(A.total_cost) AS DECIMAL (20 , 2 ))
98 FROM
99     transact A,
100    bought B
101 WHERE
102     A.transact_id = B.transact_id
103 GROUP BY HOUR(trans_time)
104 ORDER BY SUM(A.total_cost) DESC , HOUR(trans_time)
105 """

```

```

106     print(my_query)
107     cur.execute(my_query)
108     results = cur.fetchall()
109     return render_template("music6.html", results=results)
110
111
112 @app.route("/age_percentage", methods=["GET", "POST"])
113 def age_percentage():
114     cur = mysql.connection.cursor()
115     my_query = """
116 SELECT distinct
117     cast(SUM(IF(A.age < 20, 1, 0))/totals.total * 100 as decimal(20,2)) AS 'Under
118 ↵ 20',
119     cast(SUM(IF(A.age BETWEEN 20 AND 29, 1, 0))/totals.total * 100 as
120 ↵ decimal(20,2)) AS '20 - 29',
121     cast(SUM(IF(A.age BETWEEN 30 AND 39, 1, 0))/totals.total * 100 as
122 ↵ decimal(20,2)) AS '30 - 39',
123     cast(SUM(IF(A.age BETWEEN 40 AND 49, 1, 0))/totals.total * 100 as
124 ↵ decimal(20,2)) AS '40 - 49',
125     cast(SUM(IF(A.age BETWEEN 50 AND 59, 1, 0))/totals.total * 100 as
126 ↵ decimal(20,2)) AS '50 - 59',
127     cast(SUM(IF(A.age BETWEEN 60 AND 69, 1, 0))/totals.total * 100 as
128 ↵ decimal(20,2)) AS '60 - 69',
129     cast(SUM(IF(A.age >= 70, 1, 0))/totals.total * 100 as decimal (20,2)) AS 'Over
130 ↵ 70',
131     HOUR(C.trans_time) AS Hour
132 FROM
133     (SELECT distinct
134         YEAR(CURDATE()) - YEAR(birth_date) -
135 ↵ IF(STR_TO_DATE(CONCAT(YEAR(CURDATE()), '-', MONTH(birth_date), '-'),
136 ↵ DAY(birth_date)), '%Y-%c-%e') > CURDATE(), 1, 0) AS age,
137         card_id
138 FROM
139         customer) A,
140     (SELECT * from bought group by transact_id) B
141 LEFT JOIN
142     transact C on B.transact_id = C.transact_id,
143     (SELECT distinct
144         COUNT(D.transact_id) as total, HOUR(D.trans_time) as total_hour
145 FROM
146         transact D
147 GROUP BY HOUR(D.trans_time)
148 ORDER BY HOUR(D.trans_time)) totals
149 WHERE
150     A.card_id = B.card_id
151     AND B.transact_id = C.transact_id
152     AND totals.total_hour = HOUR(C.trans_time)
153 GROUP BY HOUR(C.trans_time)

```

```

145 ORDER BY HOUR(C.trans_time)
146
147 """
148     print(my_query)
149     cur.execute(my_query)
150     results = cur.fetchall()
151     return render_template("music7.html", results=results)
152
153
154 @app.route("/sales_category_store", methods=["GET", "POST"])
155 def sales_category_store():
156     cur = mysql.connection.cursor()
157     my_query = "SELECT * FROM sales_category_store"
158     print(my_query)
159     cur.execute(my_query)
160     results = cur.fetchall()
161     return render_template("music8.html", results=results)
162
163
164 @app.route("/customer_info", methods=["GET", "POST"])
165 def customer_info():
166     cur = mysql.connection.cursor()
167     my_query = "SELECT * FROM customer_info"
168     print(my_query)
169     cur.execute(my_query)
170     results = cur.fetchall()
171     return render_template("music9.html", results=results)
172
173
174 @app.route("/past_prices", methods=["GET", "POST"])
175 def past_prices():
176     cur = mysql.connection.cursor()
177     my_query = "SELECT * FROM past_prices ORDER BY time_of_change desc"
178     print(my_query)
179     cur.execute(my_query)
180     results = cur.fetchall()
181     return render_template("music10.html", results=results)
182
183
184 @app.route("/editStores")
185 def editStores():
186     return render_template("editStores.html")
187
188
189 @app.route("/search_resultS", methods=["GET", "POST"])
190 def editStoresResponce():
191     my_data = request.get_json(force=True)
192     edit = int(my_data.get("edit"))

```

```

193     print(edit)
194     if edit == -1:
195         store = my_data.get("store")
196         if store != "":
197             cur = mysql.connection.cursor()
198             my_query = "DELETE FROM store WHERE store_id = {}".format(store)
199             cur.execute(my_query)
200             cur.fetchall()
201     elif edit == 0:
202         # Update ...
203
204         data = my_data.get("phone")
205         phone = " phone = '{}'.format(data) if (data != '') else ""
206
207         data = my_data.get("city")
208         city = " city = '{}'.format(data) if (data != '') else ""
209
210         data = my_data.get("address")
211         address = " store_address = '{}'.format(data) if (data != '') else ""
212
213         data = my_data.get("postal_code")
214         postal_code = " postal_code = '{}'.format(data) if (data != '') else ""
215
216         data = my_data.get("opening_time")
217         opening_time = " opening_time = '{}'.format(data) if (data != '') else
↵ ""
218
219         data = my_data.get("closing_time")
220         closing_time = " closing_time = '{}'.format(data) if (data != '') else
↵ ""
221
222         data = my_data.get("square_meters")
223         square_meters = " square_meters = '{}'.format(data) if (data != '') else
↵ ""
224
225         additionalQuery = [
226             phone,
227             city,
228             address,
229             postal_code,
230             opening_time,
231             closing_time,
232             square_meters,
233         ]
234         additionalQuery = ",".join(list(filter(lambda i: i != "",
↵ additionalQuery)))
235     if additionalQuery != "":
236         my_query = (

```

```

237         "update store set "
238         + additionalQuery
239         + " where store_id = {};" .format(my_data.get("store"))
240     )
241
242     cur = mysql.connection.cursor()
243     cur.execute(my_query)
244     cur.fetchall()
245 elif edit == 1:
246     # Add ...
247     data = my_data.get("phone")
248     phone = ("phone", "" + data + "") if (data != "") else ""
249
250     data = my_data.get("city")
251     city = ("city", "" + data + "") if (data != "") else ""
252
253     data = my_data.get("address")
254     address = ("store_address", "" + data + "") if (data != "") else ""
255
256     data = my_data.get("postal_code")
257     postal_code = ("postal_code", "" + data + "") if (data != "") else ""
258
259     data = my_data.get("opening_time")
260     opening_time = ("opening_time", "" + data + "") if (data != "") else ""
261
262     data = my_data.get("closing_time")
263     closing_time = ("closing_time", "" + data + "") if (data != "") else ""
264
265     data = my_data.get("square_meters")
266     square_meters = ("square_meters", "" + data + "") if (data != "") else
        ↪ ""
267
268     additionalQuery = [
269         phone,
270         city,
271         address,
272         postal_code,
273         opening_time,
274         closing_time,
275         square_meters,
276     ]
277     additionalQuery = list(filter(lambda i: i != "", additionalQuery))
278     if additionalQuery != []:
279         attributes, values = zip(*additionalQuery)
280         attributes = ",".join(attributes)
281         values = ",".join(values)
282         my_query = "insert into store (" + attributes + ") values (" + values
            ↪ + ")"

```

```

283         print(my_query)
284         cur = mysql.connection.cursor()
285         cur.execute(my_query)
286         cur.fetchall()
287
288     cur = mysql.connection.cursor()
289     my_query = "select * from store"
290     cur.execute(my_query)
291     results = cur.fetchall()
292     return render_template("editStoresResponce.html", albums=results)
293
294
295 @app.route("/query_2", methods=["GET", "POST"])
296 def query_2():
297     cur = mysql.connection.cursor()
298     my_query = """
299     (SELECT
300         A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
301         ↪ A.store_id
302     FROM
303         bought A,
304         bought B,
305         has C,
306         has D
307     WHERE
308         A.transact_id = B.transact_id
309         AND A.barcode > B.barcode
310         AND A.barcode = C.barcode
311         AND A.store_id = C.store_id
312         AND B.barcode = D.barcode
313         AND B.store_id = 1
314         AND B.store_id = D.store_id
315     GROUP BY A.barcode , B.barcode, A.store_id
316     ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
317     LIMIT 1)
318     union
319     (SELECT
320         A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
321         ↪ A.store_id
322     FROM
323         bought A,
324         bought B,
325         has C,
326         has D
327     WHERE
328         A.transact_id = B.transact_id
329         AND A.barcode > B.barcode
330         AND A.barcode = C.barcode

```

```

329         AND A.store_id = C.store_id
330         AND B.barcode = D.barcode
331         AND B.store_id = 2
332         AND B.store_id = D.store_id
333     GROUP BY A.barcode , B.barcode, A.store_id
334     ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
335     LIMIT 1)
336 union
337 (SELECT
338     A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
339     ↪ A.store_id
340 FROM
341     bought A,
342     bought B,
343     has C,
344     has D
345 WHERE
346     A.transact_id = B.transact_id
347     AND A.barcode > B.barcode
348     AND A.barcode = C.barcode
349     AND A.store_id = C.store_id
350     AND B.barcode = D.barcode
351     AND B.store_id = 3
352     AND B.store_id = D.store_id
353 GROUP BY A.barcode , B.barcode, A.store_id
354 ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
355 LIMIT 1)
356 union
357 (SELECT
358     A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
359     ↪ A.store_id
360 FROM
361     bought A,
362     bought B,
363     has C,
364     has D
365 WHERE
366     A.transact_id = B.transact_id
367     AND A.barcode > B.barcode
368     AND A.barcode = C.barcode
369     AND A.store_id = C.store_id
370     AND B.barcode = D.barcode
371     AND B.store_id = 4
372     AND B.store_id = D.store_id
373 GROUP BY A.barcode , B.barcode, A.store_id
374 ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
375 LIMIT 1)
376 union

```

```

375 (SELECT
376     A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
377     ↪ A.store_id
378 FROM
379     bought A,
380     bought B,
381     has C,
382     has D
383 WHERE
384     A.transact_id = B.transact_id
385     AND A.barcode > B.barcode
386     AND A.barcode = C.barcode
387     AND A.store_id = C.store_id
388     AND B.barcode = D.barcode
389     AND B.store_id = 5
390     AND B.store_id = D.store_id
391 GROUP BY A.barcode , B.barcode, A.store_id
392 ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
393 LIMIT 1)
394 union
395 (SELECT
396     A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
397     ↪ A.store_id
398 FROM
399     bought A,
400     bought B,
401     has C,
402     has D
403 WHERE
404     A.transact_id = B.transact_id
405     AND A.barcode > B.barcode
406     AND A.barcode = C.barcode
407     AND A.store_id = C.store_id
408     AND B.barcode = D.barcode
409     AND B.store_id = 6
410     AND B.store_id = D.store_id
411 GROUP BY A.barcode , B.barcode, A.store_id
412 ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
413 LIMIT 1)
414 union
415 (SELECT
416     A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
417     ↪ A.store_id
418 FROM
419     bought A,
420     bought B,
421     has C,
422     has D

```



```

420 WHERE
421     A.transact_id = B.transact_id
422     AND A.barcode > B.barcode
423     AND A.barcode = C.barcode
424     AND A.store_id = C.store_id
425     AND B.barcode = D.barcode
426     AND B.store_id = 7
427     AND B.store_id = D.store_id
428 GROUP BY A.barcode , B.barcode, A.store_id
429 ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
430 LIMIT 1)
431 union
432 (SELECT
433     A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
434     ↪ A.store_id
435 FROM
436     bought A,
437     bought B,
438     has C,
439     has D
440 WHERE
441     A.transact_id = B.transact_id
442     AND A.barcode > B.barcode
443     AND A.barcode = C.barcode
444     AND A.store_id = C.store_id
445     AND B.barcode = D.barcode
446     AND B.store_id = 8
447     AND B.store_id = D.store_id
448 GROUP BY A.barcode , B.barcode, A.store_id
449 ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
450 LIMIT 1)
451 union
452 (SELECT
453     A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
454     ↪ A.store_id
455 FROM
456     bought A,
457     bought B,
458     has C,
459     has D
460 WHERE
461     A.transact_id = B.transact_id
462     AND A.barcode > B.barcode
463     AND A.barcode = C.barcode
464     AND A.store_id = C.store_id
465     AND B.barcode = D.barcode
466     AND B.store_id = 9
467     AND B.store_id = D.store_id

```

```

466 GROUP BY A.barcode , B.barcode, A.store_id
467 ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
468 LIMIT 1)
469 union
470 (SELECT
471     A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
472     ↪ A.store_id
473 FROM
474     bought A,
475     bought B,
476     has C,
477     has D
478 WHERE
479     A.transact_id = B.transact_id
480     AND A.barcode > B.barcode
481     AND A.barcode = C.barcode
482     AND A.store_id = C.store_id
483     AND B.barcode = D.barcode
484     AND B.store_id = 10
485     AND B.store_id = D.store_id
486 GROUP BY A.barcode , B.barcode, A.store_id
487 ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
488 LIMIT 1)
489 """
490 print(my_query)
491 cur.execute(my_query)
492 results = cur.fetchall()
493 return render_template("query_2.html", results=results)
494
495 @app.route("/query_1", methods=["GET", "POST"])
496 def query_1():
497     cur = mysql.connection.cursor()
498     my_query = """
499 (select p.category, h.store_id, COUNT(*) as counter
500 FROM product p, has h, bought b
501 WHERE b.store_id = h.store_id
502 AND h.barcode = b.barcode
503 AND h.barcode = p.barcode
504 AND p.category = 'Φρέσκα Προϊόντα'
505 GROUP BY h.store_id
506 ORDER BY counter desc
507 limit 1)
508 UNION
509 (select p.category, h.store_id, COUNT(*) as counter
510 FROM product p, has h, bought b
511 WHERE b.store_id = h.store_id
512 AND h.barcode = b.barcode

```

```

513 AND h.barcode = p.barcode
514 AND p.category = 'Ποϊόντα Ψυγείου'
515 GROUP BY h.store_id
516 ORDER BY counter desc
517 limit 1)
518 UNION
519 (select p.category, h.store_id, COUNT(*) as counter
520 FROM product p, has h, bought b
521 WHERE b.store_id = h.store_id
522 AND h.barcode = b.barcode
523 AND h.barcode = p.barcode
524 AND p.category = 'Κατοικίδια'
525 GROUP BY h.store_id
526 ORDER BY counter desc
527 limit 1)
528 UNION
529 (select p.category, h.store_id, COUNT(*) as counter
530 FROM product p, has h, bought b
531 WHERE b.store_id = h.store_id
532 AND h.barcode = b.barcode
533 AND h.barcode = p.barcode
534 AND p.category = 'Κάβα'
535 GROUP BY h.store_id
536 ORDER BY counter desc
537 limit 1)
538 UNION
539 (select p.category, h.store_id, COUNT(*) as counter
540 FROM product p, has h, bought b
541 WHERE b.store_id = h.store_id
542 AND h.barcode = b.barcode
543 AND h.barcode = p.barcode
544 AND p.category = 'Είδη Σπιτιού'
545 GROUP BY h.store_id
546 ORDER BY counter desc
547 limit 1)
548 UNION
549 (select p.category, h.store_id, COUNT(*) as counter
550 FROM product p, has h, bought b
551 WHERE b.store_id = h.store_id
552 AND h.barcode = b.barcode
553 AND h.barcode = p.barcode
554 AND p.category = 'Προσωπικής Περιποίησης'
555 GROUP BY h.store_id
556 ORDER BY counter desc
557 limit 1)
558 UNION
559 (select p.category, h.store_id, COUNT(*) as counter
560 FROM product p, has h, bought b

```

```

561 WHERE b.store_id = h.store_id
562 AND h.barcode = b.barcode
563 AND h.barcode = p.barcode
564 AND p.category = 'Είδη Σπιτιού'
565 GROUP BY h.store_id
566 ORDER BY counter desc
567 limit 1)
568 """
569     print(my_query)
570     cur.execute(my_query)
571     results = cur.fetchall()
572     return render_template("query_1.html", results=results)
573
574
575 @app.route("/editProducts")
576 def editProducts():
577     return render_template("editProducts.html")
578
579
580 @app.route("/search_resultP", methods=["GET", "POST"])
581 def editProductsResponse():
582     my_data = request.get_json(force=True)
583     edit = int(my_data.get("action"))
584     print(edit)
585     if edit == -1:
586         barcode = my_data.get("barcode")
587         if barcode != "":
588             cur = mysql.connection.cursor()
589             my_query = "DELETE FROM product WHERE barcode = {}".format(barcode)
590             cur.execute(my_query)
591             mysql.connection.commit()
592     elif edit == 0:
593         # Update ...
594
595         data = my_data.get("product_name")
596         product_name = " product_name = '{}'.format(data) if (data != "") else
↵      ""
597
598         data = my_data.get("producer_name")
599         producer_name = " producer_name = '{}'.format(data) if (data != "") else
↵      ""
600
601         data = my_data.get("price")
602         price = " price = '{}'.format(data) if (data != "") else ""
603
604         data = my_data.get("special_note")
605         special_note = " special_note = b'{}'.format(data) if (data != "") else
↵      ""

```

```

606
607 data = my_data.get("category")
608 category = " category = '{}'.format(data) if (data != "") else ""
609
610 additionalQuery = [
611     product_name,
612     producer_name,
613     price,
614     special_note,
615     category,
616 ]
617 additionalQuery = ",".join(list(filter(lambda i: i != "",
↵ additionalQuery)))
618 if additionalQuery != "":
619     my_query = (
620         "update product set "
621         + additionalQuery
622         + " where barcode = {};".format(my_data.get("barcode"))
623     )
624
625     cur = mysql.connection.cursor()
626     cur.execute(my_query)
627     mysql.connection.commit()
628 elif edit == 1:
629     # Add ...
630     data = my_data.get("barcode")
631     barcode = ("barcode", "" + data + "") if (data != "") else ""
632
633     data = my_data.get("product_name")
634     product_name = ("product_name", "" + data + "") if (data != "") else ""
635
636     data = my_data.get("producer_name")
637     producer_name = ("producer_name", "" + data + "") if (data != "") else
↵ ""
638
639     data = my_data.get("price")
640     price = ("price", "" + data + "") if (data != "") else ""
641
642     data = my_data.get("special_note")
643     special_note = ("special_note", "b'" + data + "'") if (data != "") else ""
644
645     data = my_data.get("category")
646     category = ("category", "" + data + "") if (data != "") else ""
647
648     additionalQuery = [
649         barcode,
650         product_name,
651         producer_name,

```

```

652         price,
653         special_note,
654         category,
655     ]
656     additionalQuery = list(filter(lambda i: i != "", additionalQuery))
657     if additionalQuery != []:
658         attributes, values = zip(*additionalQuery)
659         attributes = ",".join(attributes)
660         values = ",".join(values)
661         my_query = (
662             "insert into product (" + attributes + ") values (" + values + ")"
663         )
664         print(my_query)
665         cur = mysql.connection.cursor()
666         cur.execute(my_query)
667         mysql.connection.commit()
668
669     cur = mysql.connection.cursor()
670     my_query = "select * from product"
671     cur.execute(my_query)
672     results = cur.fetchall()
673     return render_template("editProductsResponce.html", albums=results)
674
675
676 @app.route("/editCustomers")
677 def editCustomers():
678     return render_template("editCustomers.html")
679
680
681 @app.route("/search_resultC", methods=["GET", "POST"])
682 def editCustomersResponce():
683     my_data = request.get_json(force=True)
684     edit = int(my_data.get("edit"))
685     print(edit)
686     if edit == -1:
687         card_id = my_data.get("card_id")
688         if card_id != "":
689             cur = mysql.connection.cursor()
690             my_query = "DELETE FROM customer WHERE card_id = {}".format(card_id)
691             cur.execute(my_query)
692             mysql.connection.commit()
693     elif edit == 0:
694         card_id = my_data.get("card_id")
695         if card_id != "":
696             # Update ...
697
698         data = my_data.get("first_name")

```

```

699 first_name = " first_name = '{} ' ".format(data) if (data != "") else
    ↳ ""
700
701 data = my_data.get("last_name")
702 last_name = " last_name = '{} ' ".format(data) if (data != "") else ""
703
704 data = my_data.get("total_points")
705 total_points = " total_points = '{} ' ".format(data) if (data != "")
    ↳ else ""
706
707 data = my_data.get("rewards")
708 rewards = " rewards = '{} ' ".format(data) if (data != "") else ""
709
710 data = my_data.get("birthdate")
711 birthdate = " birth_date = '{} ' ".format(data) if (data != "") else ""
712
713 data = my_data.get("marital_status")
714 marital_status = (
715     " marital_status = '{} ' ".format(data) if (data != "") else ""
716 )
717
718 data = my_data.get("social_security_number")
719 social_security_number = (
720     " social_security_number = '{} ' ".format(data) if (data != "")
    ↳ else ""
721 )
722
723 phone_number = my_data.get("phone_number")
724 if phone_number != "":
725     editNumber = int(my_data.get("editNumber"))
726     phone_number = phone_number.split(",")
727     if editNumber == -1:
728         my_query = "DELETE FROM customer_phone WHERE card_id = {} and
    ↳ phone_number in ({});".format(
729             card_id, ",".join(phone_number)
730         )
731         cur = mysql.connection.cursor()
732         cur.execute(my_query)
733         mysql.connection.commit()
734     elif editNumber == 1:
735         for phone in phone_number:
736             my_query = "insert into customer_phone
    ↳ (card_id,phone_number) values ('{}','{}');".format(
737                 card_id, phone
738             )
739             cur = mysql.connection.cursor()
740             cur.execute(my_query)
741             mysql.connection.commit()

```

```

742
743         additionalQuery = [
744             first_name,
745             last_name,
746             total_points,
747             rewards,
748             birthdate,
749             marital_status,
750             social_security_number,
751         ]
752         additionalQuery = ",".join(list(filter(lambda i: i != "",
753             ↪ additionalQuery)))
754         if additionalQuery != "":
755             my_query = (
756                 "update customer set "
757                 + additionalQuery
758                 + " where card_id = {};" .format(my_data.get("card_id"))
759             )
760             cur = mysql.connection.cursor()
761             cur.execute(my_query)
762             mysql.connection.commit()
763     elif edit == 1:
764         # Add ...
765         data = my_data.get("first_name")
766         first_name = ("first_name", "" + data + "") if (data != "") else ""
767
768         data = my_data.get("last_name")
769         last_name = ("last_name", "" + data + "") if (data != "") else ""
770
771         data = my_data.get("total_points")
772         total_points = ("total_points", "" + data + "") if (data != "") else ""
773
774         data = my_data.get("rewards")
775         rewards = ("rewards", "" + data + "") if (data != "") else ""
776
777         data = my_data.get("birth_date")
778         birth_date = ("birth_date", "" + data + "") if (data != "") else ""
779
780         data = my_data.get("marital_status")
781         marital_status = ("marital_status ", "" + data + "") if (data != "")
782             ↪ else ""
783
784         data = my_data.get("social_security_number")
785         social_security_number = (
786             ("social_security_number", "" + data + "") if (data != "") else ""
787         )

```



```

788     additionalQuery = [
789         first_name,
790         last_name,
791         total_points,
792         rewards,
793         birth_date,
794         marital_status,
795         social_security_number,
796     ]
797     additionalQuery = list(filter(lambda i: i != "", additionalQuery))
798     if additionalQuery != []:
799         attributes, values = zip(*additionalQuery)
800         attributes = ",".join(attributes)
801         values = ",".join(values)
802         my_query = (
803             "insert into customer (" + attributes + ") values (" + values +
804             "↵ ")"
805         )
806         print(my_query)
807         cur = mysql.connection.cursor()
808         cur.execute(my_query)
809         mysql.connection.commit()
810
811     phone_number = my_data.get("phone_number")
812     if phone_number != "":
813         editNumber = int(my_data.get("editNumber"))
814         phone_number = phone_number.split(",")
815         if editNumber == 1:
816             card = "select max(card_id) from customer;"
817             cur = mysql.connection.cursor()
818             cur.execute(card)
819             card_id = cur.fetchall()[0][0]
820             mysql.connection.commit()
821             for phone in phone_number:
822                 my_query = "insert into customer_phone
823                 ↵ (card_id,phone_number) values ('{}','{}');".format(
824                     card_id, phone
825                 )
826                 print(my_query)
827                 cur = mysql.connection.cursor()
828                 cur.execute(my_query)
829                 mysql.connection.commit()
830
831     cur = mysql.connection.cursor()
832     my_query = "select A.*, B.phone_number from customer A, customer_phone B where
833     ↵ B.card_id = A.card_id;"
834     cur.execute(my_query)
835     results = cur.fetchall()

```

```

833     return render_template("editCustomersResponce.html", albums=results)
834
835
836 @app.route("/search")
837 def search():
838     return render_template("index.html")
839
840
841 @app.route("/search_result", methods=["GET", "POST"])
842 def search_result():
843     my_data = request.get_json(force=True)
844     """
845     my_data.get("store") = "all stores"
846     my_data.get("birthday") = ""
847     my_data.get("quantity") = ""
848     my_data.get("barcode") = ""
849     my_data.get("total_amount") = ""
850     my_data.get("payment_method") = "either"
851     my_data.get("category") = "any'
852
853
854     """
855     storeid = (
856         " and bought.store_id = {}".format(my_data.get("store"))
857         if (my_data.get("store") != "")
858         else ""
859     )
860     transdate = (
861         " and month(transact.trans_date) = month('{0}')" and
862         ↪ year(transact.trans_date) = year('{0}').format(
863             my_data.get("birthday")
864         )
865         if (my_data.get("birthday") != "")
866         else ""
867     )
868     quantity = (
869         " and bought.quantity >= {}".format(my_data.get("quantity"))
870         if (my_data.get("quantity") != "")
871         else ""
872     )
873     barcode = (
874         " and bought.barcode = {}".format(my_data.get("barcode"))
875         if (my_data.get("barcode") != "")
876         else ""
877     )
878     total_cost = (
879         " and transact.total_cost >= {}".format(my_data.get("total_amount"))
880         if (my_data.get("total_amount") != "")

```

```

880         else ""
881     )
882     payment_method = (
883         " and transact.payment_method =
884         ↳ '{}'.format(my_data.get("payment_method"))
885         if (my_data.get("payment_method") != "")
886         else ""
887     )
888     category = (
889         " and bought.barcode in (select barcode from product where category =
890         ↳ '{}'.format(
891             my_data.get("category")
892         )
893         if (my_data.get("category") != "")
894         else ""
895     )
896     # print("Here:", storeid)
897     everything = "select transact.transact_id,transact.total_cost,
898     ↳ transact.trans_date,transact.trans_time,transact.week_day,transact.payment_method,bought.c
899     ↳ from transact, bought where bought.transact_id = transact.transact_id"
900     cur = mysql.connection.cursor()
901     my_query = (
902         everything
903         + storeid
904         + transdate
905         + quantity
906         + barcode
907         + total_cost
908         + payment_method
909         + category
910         + " limit 1000"
911     )
912     print(my_query)
913     cur.execute(my_query)
914     results = cur.fetchall()
915     return render_template("music.html", albums=results)
916
917 @app.route("/customers", methods=["GET", "POST"])
918 def customerNew():
919     my_data = request.get_json()
920     actual_data = 0
921     if my_data == None:
922         actual_data = 0
923     else:
924         actual_data = int(my_data.get("store"))
925     cur = mysql.connection.cursor()
926     myquery4 = """select hour(trans_time), count(hour(trans_time))

```

```

924         from transact
925         where transact_id in (
926             select transact_id
927             from bought
928             where card_id = {}
929         )
930         group by hour(trans_time)
931         order by hour(trans_time)""".format(
932     actual_data
933 )
934     cur.execute(myquery4)
935     result = []
936     x = ("x", "y")
937     for row in cur:
938         result.append(dict(zip(x, row)))
939     print(result)
940     return render_template("customers.html", result=result)
941
942
943 @app.route("/customers_visit_data")
944 def customers():
945     return render_template("customers_visit_data.html")
946
947
948 @app.route("/customer_result", methods=["GET", "POST"])
949 def customer_result():
950     my_data = request.get_json(force=True)
951     my_useful_data = my_data["card_id"]
952     cur = mysql.connection.cursor()
953     my_query_2 = "SELECT distinct b.card_id, b.store_id FROM bought as b, store as
954     ↪ s where b.store_id = s.store_id AND b.card_id = {} order by card_id,
955     ↪ store_id".format(
956         my_useful_data
957     )
958     print(my_query_2)
959     cur.execute(my_query_2)
960     results2 = cur.fetchall()
961     my_query_3 = """
962     SELECT card_id, barcode, sum(quantity)
963     FROM bought
964     where card_id = {}
965     group by barcode
966     order by sum(quantity) desc
967     limit 10
968     """.format(
969         my_useful_data
970     )
971     print(my_query_3)

```

```

970     cur.execute(my_query_3)
971     results3 = cur.fetchall()
972     my_query_4 = """
973     SELECT
974         b.card_id, MONTHNAME(t.trans_date), cast(AVG(t.total_cost) as decimal(6,2))
975 FROM
976     transact AS t,
977     bought AS b
978 WHERE
979     b.card_id = {}
980     AND b.transact_id = t.transact_id
981 GROUP BY MONTHNAME(t.trans_date)
982 ORDER BY MONTH(t.trans_date);
983 """.format(
984     my_useful_data
985 )
986 print(my_query_4)
987 cur.execute(my_query_4)
988 results4 = cur.fetchall()
989 my_query_5 = """
990 SELECT
991     b.card_id,
992     WEEK(t.trans_date) Week,
993     cast(AVG(total_cost) as decimal(6,2))
994 FROM
995     bought AS b,
996     transact AS t
997 WHERE
998     b.card_id = {}
999     AND b.transact_id = t.transact_id
1000 GROUP BY WEEK(t.trans_date)
1001 """.format(
1002     my_useful_data
1003 )
1004 print(my_query_5)
1005 cur.execute(my_query_5)
1006 results5 = cur.fetchall()
1007
1008 # the_id = request.args.get('button_id')
1009 return render_template(
1010     "music2.html",
1011     results=results2,
1012     results3=results3,
1013     results4=results4,
1014     results5=results5,
1015 )
1016
1017

```

```

1018 @app.route("/report2", methods=["GET", "POST"])
1019 def report2():
1020     abduction = {
1021         "firstname": request.form["firstname"],
1022         "lastname": request.form["lastname"],
1023     }
1024     return render_template("report2.html", abduction=abduction)
1025
1026
1027 # start the server
1028 if __name__ == "__main__":
1029     app.run(
1030         host=os.getenv("IP", "0.0.0.0"), port=int(os.getenv("PORT", 8587)),
1031         ↪ debug=True
1032     )

```

## 4.3 FrontEnd Server

- [JavaScript](#)
- [CSS](#)
- [Fonts](#)
- [HTML](#)

## 5 Video Indexing

| Question | Start | End   |
|----------|-------|-------|
| 1        | 00:02 | 00:35 |
| 2        | 00:36 | 01:16 |
| 3        | 01:17 | 05:25 |
| 4        | 05:26 | 05:43 |
| 5        | 05:44 | 06:15 |
| 6        | 06:16 | 07:14 |
| 7        | 07:15 | 08:06 |
| 8        | 08:07 | 08:22 |
| 9        | 08:23 | 08:38 |
| 10       | 08:39 | 08:48 |
| 11       | 08:49 | 09:08 |
| 12       | 09:09 | 09:25 |
| 13       | 09:26 | 10:09 |
| 14       | 10:10 | 10:51 |
| 15       | 10:52 | 11:54 |

## 6 Links

- [Our Site](#)
- [Github Repository](#)
- [YouTube Video](#)