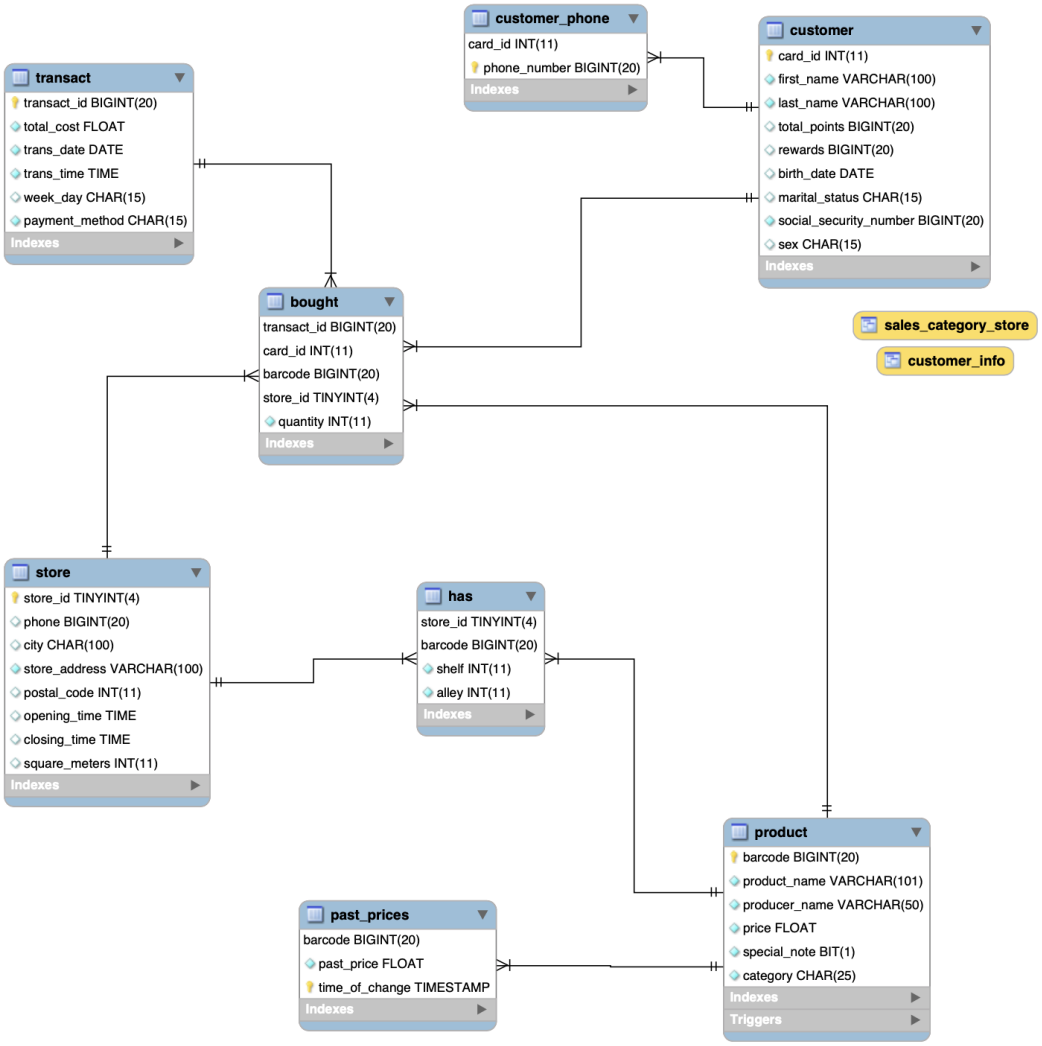
Team Project in Databases

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

June 2020

Contents

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

```
use AlexJohnChris;
CREATE INDEX hour_trans ON transact(time);
CREATE INDEX special_bit ON product(special_note);
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:

```
SELECT
   A.category,
   A.special_note,
   COUNT(B.barcode) / COUNT(C.barcode) * 100 AS percentage
FROM
   product A,
   bought B,
   product C
WHERE
A.barcode = B.barcode
   AND B.barcode = C.barcode
   AND A.special_note = 1
GROUP BY A.special_note , A.category
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 folowing command in terminal, using your credentials in order to connect in mysql host:

```
$ 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,

```
$ git clone https://github.com/AlexandrosKyriakakis/DataBase.git
$ cd DataBase
$ 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,

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

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

```
$ python3 server_guest.py
```

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

2 DDL

```
CREATE DATABASE `AlexJohnChris`;
ALTER DATABASE AlexJohnChris CHARACTER
SET utf8
COLLATE utf8_bin;
USE `AlexJohnChris`;
/* create table Store*/
CREATE TABLE store
(
 store_id TINYINT NOT NULL
 AUTO_INCREMENT,
    phone BIGINT,
    city CHAR
 (100),
    store_address VARCHAR
 (100) NOT NULL,
    postal_code INT,
                opening_time TIME,
    closing_time TIME,
    square_meters INT,
    CONSTRAINT PKstore PRIMARY KEY
 (store_id));

 /* create table Customer*/
 CREATE TABLE customer
 (
  card_id INT NOT NULL
  AUTO_INCREMENT,
   first_name VARCHAR
  (100) NOT NULL,
   last_name VARCHAR
  (100) NOT NULL,
   total_points BIGINT DEFAULT 0,
     rewards BIGINT DEFAULT 0,
   birth_date DATE,
   marital_status CHAR
  (15),
   social_security_number BIGINT NOT NULL,
   sex CHAR
  (15),
   CONSTRAINT PKcustomer PRIMARY KEY
  (card_id));

  /*customer_phone multivalued attribute of customer*/
  CREATE TABLE customer_phone
  (
   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 FK1store 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
```

```
where card_id = 99
      )
group by hour(trans_time)
order by hour(trans_time)
```

```
SELECT
   distinct b.card_id, b.store_id
FROM
   bought as b, store as s
where
   b.store_id = s.store_id
   AND b.card_id = 99
order by card_id, store_id

SELECT card_id, barcode, sum(quantity)
FROM bought
where card_id = 99
group by barcode
order by sum(quantity) desc
limit 10


SELECT
    b
.card_id, MONTHNAME
(t.trans_date),
    cast
(AVG
(t.total_cost) as decimal
(6,2))
FROM
    transact AS t,
    bought AS b
WHERE
   b.card_id = 99
   AND b.transact_id = t.transact_id
GROUP BY MONTHNAME
(t.trans_date)
ORDER BY MONTH
(t.trans_date);


SELECT
   b.card_id,
   WEEK(t.trans_date) Week,
   cast(AVG(total_cost) as decimal(6,2))
FROM
   bought AS b,
```

```
   transact AS t
WHERE
   b.card_id = 99
   AND b.transact_id = t.transact_id
GROUP BY WEEK(t.trans_date)
```

3.2 Edit Data

```
insert into store
   (phone,city,store_address,postal_code,
   opening_time,closing_time ,square_meters)
values
   ('1111111', 'Test', 'Test', '1111',
      '08:00:00', '21:00:00', '1111')
```

```
DELETE FROM store WHERE store_id = 11
```

```
update store
set
   phone = '222222' , city = 'NewTest' ,
   store_address = 'NewTest' , postal_code = '22222',
   opening_time = '09:00:00' , closing_time = '22:00:00' ,
   square_meters = '22222'  where store_id = 11;
```

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

3.3 Indexes

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

3.4 Product Data

```
SELECT distinct
   cast(SUM(
IF(A.age < 20, 1, 0))/totals.total _ 100 as decimal
(20,2)) AS 'Under 20',
   cast
(SUM
(
IF(A.age BETWEEN 20 AND 29, 1, 0))/totals.total _ 100 as decimal
(20,2)) AS '20 - 29',
   cast
(SUM
(
IF(A.age BETWEEN 30 AND 39, 1, 0))/totals.total _ 100 as decimal
(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(),
              ↪  1, 0) AS age,
42            card_id
43      FROM
44            customer) A,
45  (SELECT _
46  from bought
47  group by transact_id)
48  B
49  LEFT JOIN
50      transact C on B.transact_id = C.transact_id,
51  (SELECT distinct
52      COUNT(D.transact_id) as total, HOUR(D.trans_time) as total_hour
53  FROM
54      transact D
55  GROUP BY HOUR(D.trans_time)
56  ORDER BY HOUR(D.trans_time)
57  ) totals
58  WHERE
59      A.card_id = B.card_id
60      AND B.transact_id = C.transact_id
61      AND totals.total_hour = HOUR
```

```
(C.trans_time)
GROUP BY HOUR
(C.trans_time)
ORDER BY HOUR
(C.trans_time)
```

```
(select p.category, h.store*id, COUNT(*) as counter
FROM product p, has h, bought b
WHERE b.store*id = h.store_id
AND h.barcode = b.barcode
AND h.barcode = p.barcode
AND p.category = 'Φρέσκα Προϊόντα'
GROUP BY h.store_id
ORDER BY counter desc
limit 1)
UNION
(select p.category, h.store_id, COUNT(*) as counter
FROM product p, has h, bought b
WHERE b.store*id = h.store_id
AND h.barcode = b.barcode
AND h.barcode = p.barcode
AND p.category = 'Ποϊόντα Ψυγείου'
GROUP BY h.store_id
ORDER BY counter desc
limit 1)
UNION
(select p.category, h.store_id, COUNT(*) as counter
FROM product p, has h, bought b
WHERE b.store*id = h.store_id
AND h.barcode = b.barcode
AND h.barcode = p.barcode
AND p.category = 'Κατοικίδια'
GROUP BY h.store_id
ORDER BY counter desc
limit 1)
UNION
(select p.category, h.store_id, COUNT(*) as counter
FROM product p, has h, bought b
WHERE b.store*id = h.store_id
AND h.barcode = b.barcode
AND h.barcode = p.barcode
AND p.category = 'Κάβα'
GROUP BY h.store_id
ORDER BY counter desc
limit 1)
UNION
(select p.category, h.store_id, COUNT(*) as counter
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(\*) 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
```

```
A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
FROM
bought A,
bought B,
has C,
has D
WHERE
A.transact_id = B.transact_id
AND A.barcode > B.barcode
AND A.barcode = C.barcode
AND A.store_id = C.store_id
AND B.barcode = D.barcode
AND B.store_id = 2
AND B.store_id = D.store_id
GROUP BY A.barcode , B.barcode, A.store_id
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 1)
union
(SELECT
A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
FROM
bought A,
bought B,
has C,
has D
WHERE
A.transact_id = B.transact_id
AND A.barcode > B.barcode
AND A.barcode = C.barcode
AND A.store_id = C.store_id
AND B.barcode = D.barcode
AND B.store_id = 3
AND B.store_id = D.store_id
GROUP BY A.barcode , B.barcode, A.store_id
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 1)
union
(SELECT
A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
FROM
bought A,
bought B,
has C,
has D
WHERE
A.transact_id = B.transact_id
AND A.barcode > B.barcode
AND A.barcode = C.barcode
```

```
AND A.store_id = C.store_id
AND B.barcode = D.barcode
AND B.store_id = 4
AND B.store_id = D.store_id
GROUP BY A.barcode , B.barcode, A.store_id
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 1)
union
(SELECT
A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
FROM
bought A,
bought B,
has C,
has D
WHERE
A.transact_id = B.transact_id
AND A.barcode > B.barcode
AND A.barcode = C.barcode
AND A.store_id = C.store_id
AND B.barcode = D.barcode
AND B.store_id = 5
AND B.store_id = D.store_id
GROUP BY A.barcode , B.barcode, A.store_id
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 1)
union
(SELECT
A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
FROM
bought A,
bought B,
has C,
has D
WHERE
A.transact_id = B.transact_id
AND A.barcode > B.barcode
AND A.barcode = C.barcode
AND A.store_id = C.store_id
AND B.barcode = D.barcode
AND B.store_id = 6
AND B.store_id = D.store_id
GROUP BY A.barcode , B.barcode, A.store_id
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 1)
union
(SELECT
A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
```

```
FROM
bought A,
bought B,
has C,
has D
WHERE
A.transact_id = B.transact_id
AND A.barcode > B.barcode
AND A.barcode = C.barcode
AND A.store_id = C.store_id
AND B.barcode = D.barcode
AND B.store_id = 7
AND B.store_id = D.store_id
GROUP BY A.barcode , B.barcode, A.store_id
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 1)
union
(SELECT
A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
FROM
bought A,
bought B,
has C,
has D
WHERE
A.transact_id = B.transact_id
AND A.barcode > B.barcode
AND A.barcode = C.barcode
AND A.store_id = C.store_id
AND B.barcode = D.barcode
AND B.store_id = 8
AND B.store_id = D.store_id
GROUP BY A.barcode , B.barcode, A.store_id
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 1)
union
(SELECT
A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
FROM
bought A,
bought B,
has C,
has D
WHERE
A.transact_id = B.transact_id
AND A.barcode > B.barcode
AND A.barcode = C.barcode
AND A.store_id = C.store_id
```

```
AND B.barcode = D.barcode
AND B.store_id = 9
AND B.store_id = D.store_id
GROUP BY A.barcode , B.barcode, A.store_id
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 1)
union
(SELECT
A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode), A.store_id
FROM
bought A,
bought B,
has C,
has D
WHERE
A.transact_id = B.transact_id
AND A.barcode > B.barcode
AND A.barcode = C.barcode
AND A.store_id = C.store_id
AND B.barcode = D.barcode
AND B.store_id = 10
AND B.store_id = D.store_id
GROUP BY A.barcode , B.barcode, A.store_id
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 1)
```

```
SELECT
   A.barcode, B.barcode, COUNT(A.barcode)
FROM
   bought A,
   bought B
WHERE
   A.transact_id = B.transact_id
   AND A.barcode > B.barcode
GROUP BY A.barcode , B.barcode
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 100;
```

```
SELECT
   A.store_id, A.shelf, A.alley, COUNT(B.barcode)
FROM
   has A,
   bought B
WHERE
   A.barcode = B.barcode
GROUP BY B.barcode , A.store_id , A.shelf , A.alley
ORDER BY COUNT(B.barcode) DESC
LIMIT 100;
```

```sql
SELECT
   HOUR(trans_time),
   CAST(SUM(A.total_cost) AS DECIMAL (20 , 2 ))
FROM
   transact A,
   bought B
WHERE
   A.transact_id = B.transact_id
GROUP BY HOUR(trans_time)
ORDER BY SUM(A.total_cost) DESC , HOUR(trans_time)
```

```sql
SELECT
   A.category,
   A.special_note,
   COUNT(B.barcode) / COUNT(C.barcode) * 100 AS percentage
FROM
   product A,
   bought B,
   product C
WHERE
   A.barcode = B.barcode
   AND B.barcode = C.barcode
   AND A.special_note = 1
GROUP BY A.special_note , A.category
ORDER BY percentage DESC;
```

3.5 Search Per Condition

```sql
use AlexJohnChris;
select
   transact.transact_id, transact.total_cost,
   transact.trans_date, transact.trans_time,
   transact.week_day, transact.payment_method,
   bought.card_id, bought.barcode, bought.quantity
from transact, bought
where
   bought.transact_id = transact.transact_id
   and bought.store_id = 2
   and month(transact.trans_date) = month('1980-11-01')
   and year(transact.trans_date) = year('1980-11-01')
   and bought.quantity >= 4
   and bought.barcode = 5000665
   and transact.total_cost >= 3.0
   and transact.payment_method = 'Card'
   and bought.barcode in (select barcode
   from product
   where category = 'Είδη Σπιτιού')
limit 1000;
```

3.6 Add Stores

```
USE `AlexJohnChris`;
INSERT INTO store
    (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
        VALUES (2108476357,'Αθήνα','Κουμουνδούρου & Παλαιολόγου
        46-48',17937,'08:00:00','21:00:00',240);

INSERT INTO store
    (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
        VALUES (2108476357,'Καρπενήσι','Αγ.
        Γεώργιος',36071,'08:00:00','21:00:00',292);

INSERT INTO store
    (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
        VALUES (2108476357,'Αθήνα','Θηβών-Ιδομενέως & Ανωνύμου
        Οδού',13121,'08:00:00','21:00:00',372);

INSERT INTO store
    (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
        VALUES (2108476357,'Αθήνα','Λ. Ηλιουπόλεως 4 & Κασσιόπης 3
        ',17237,'08:00:00','21:00:00',534);

INSERT INTO store
    (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
        VALUES (2108476357,'Αθήνα','Φιλαδελφείας
        101',13671,'08:00:00','21:00:00',4234);

INSERT INTO store
    (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
        VALUES (2108476357,'Αθήνα','Πεύκων
        26',15126,'08:00:00','21:00:00',423);

INSERT INTO store
    (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
        VALUES (2108476357,'Χαλκίδα','NEA
        ΑΡΤΑΚΗ',34600,'08:00:00','21:00:00',123);

INSERT INTO store
    (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
        VALUES (2108476357,'Χαλκίδα','Γ. Παπανδρέου & ΄Ανδρου
        1',34100,'08:00:00','21:00:00',45);

INSERT INTO store
    (phone,city,store_address,postal_code,opening_time,closing_time,square_meters)
        VALUES (2108476357,'Αθήνα','Μακεδονίας 2 & Τενέδου
        8',15235,'08:00:00','21:00:00',3124);

```

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

3.7 Trigers

```
use AlexJohnChris;
DELIMITER //
DROP TRIGGER IF EXISTS past_p
//
CREATE TRIGGER past_p AFTER
UPDATE
ON product
FOR EACH ROW
BEGIN
   IF (not NEW.price <=> OLD.price) THEN
   INSERT INTO past_prices
      (barcode,past_price,time_of_change)
   VALUES
      (OLD.barcode, OLD.price, NOW());
END
IF;
END;//
DELIMITER ;
```

3.8 Views

```
use AlexJohnChris;
CREATE VIEW sales_category_store
AS
SELECT B.transact_id, B.card_id, H.barcode, B.quantity, P.category, H.store_id
FROM
bought B, has H, product P
WHERE B.barcode = H.barcode
AND H.barcode = P.barcode
ORDER BY P.category, H.store_id
```

```
use AlexJohnChris;
CREATE VIEW customer_info
AS
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
FROM
customer c
LEFT JOIN
customer_phone p
on c.card_id = p.card_id
```

4 Installation Code

4.1 Add Data

```
import mysql.connector
import pandas as pd
import numpy as np
People = pd.read_csv("./MarketDataset/data/RandomPeople.csv")
mydb = mysql.connector.connect(
    host = "****",
    user = "****",
    passwd = "****",
    database = "AlexJohnChris"
)

mycursor = mydb.cursor()
rewardAll = np.arange(1,100,1)
pointsAll = np.arange(1,1000,1)
phones = np.arange(2100000000,2109999999,10)
for i in range(len(People)):
#   ssn,first_name,last_name,email,gender,birth
    first_name = People['first_name'][i].replace("'","").replace('"',"")
    last_name = People['last_name'][i].replace("'","").replace('"',"")
    email = People['email'][i]
    sex = People['gender'][i]
    total_points = np.random.choice(pointsAll)
    rewards = np.random.choice(rewardAll)
    mar = 'Widow' if sex == 'Female' else 'Widower'
    social_security_number = People['ssn'][i].replace('-','')
    birth_date = People['birth'][i]
    marital_status =
    ↪  np.random.choice(['Married','Single','Divorced','Engaged',mar])
    sqlFormula = """INSERT INTO customer
    ↪  (first_name,last_name,birth_date,marital_status,sex,social_security_number,total_points,re
    ↪
                    VALUES
↪  ('{}','{}','{}','{}','{}',{},{},{})""".format(first_name,last_name,birth_date,marital_status,s
    mycursor.execute(sqlFormula)
    mydb.commit() #Save Data
    for k in range(3):
        sqlFormula = """INSERT INTO customer_phone (card_id,phone_number)
                        VALUES ({},{})""".format(i+1,np.random.choice(phones))
        mycursor.execute(sqlFormula)
    mydb.commit() #Save Data
```

```
import mysql.connector
import pandas as pd
import numpy as np
People = pd.read_csv("./MarketDataset/data/RandomPeople.csv")
```

```
mydb = mysql.connector.connect(
    host = "****",
    user = "****",
    passwd = "****",
    database = "AlexJohnChris"
)

mycursor = mydb.cursor()
rewardAll = np.arange(1,100,1)
pointsAll = np.arange(1,1000,1)
phones = np.arange(2100000000,2109999999,10)
for i in range(len(People)):
#   ssn,first_name,last_name,email,gender,birth
    first_name = People['first_name'][i].replace("'","").replace('"',"")
    last_name = People['last_name'][i].replace("'","").replace('"',"")
    email = People['email'][i]
    sex = People['gender'][i]
    total_points = np.random.choice(pointsAll)
    rewards = np.random.choice(rewardAll)
    mar = 'Widow' if sex == 'Female' else 'Widower'
    social_security_number = People['ssn'][i].replace('-','')
    birth_date = People['birth'][i]
    marital_status =
    ↪  np.random.choice(['Married','Single','Divorced','Engaged',mar])
    sqlFormula = """INSERT INTO customer
    ↪  (first_name,last_name,birth_date,marital_status,sex,social_security_number,total_points,re
    ↪
                    VALUES
↪  ('{}','{}','{}','{}','{}',{},{},{})""".format(first_name,last_name,birth_date,marital_status,s
    mycursor.execute(sqlFormula)
    mydb.commit() #Save Data
    for k in range(3):
        sqlFormula = """INSERT INTO customer_phone (card_id,phone_number)
                        VALUES ({},{})""".format(i+1,np.random.choice(phones))
        mycursor.execute(sqlFormula)
    mydb.commit() #Save Data
```

```
import mysql.connector
import random as rd
import pandas as pd
import numpy as np
ab = pd.read_csv("./MarketDataset/data/AB.csv")
mydb = mysql.connector.connect(
    host = "****",
    user = "****",
    passwd = "****",
    database = "AlexJohnChris"
)
```

```
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 =
        ↪  "-".join([str(np.random.choice(year)),str(np.random.choice(month)),
        ↪  str(np.random.choice(day))])
22      trans_time =
        ↪  ":".join([str(np.random.choice(hour)),str(np.random.choice(minute)),
        ↪  str(np.random.choice(second))])
23      payment_method = np.random.choice(['Cash','Card'])
24      week_day =
        ↪  np.random.choice(['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'])
25      sqlFormula = """INSERT INTO transact
        ↪  (total_cost,trans_date,trans_time,week_day,payment_method)
26                      VALUES
    ↪  ({},'{}','{}','{}','{}')""".format(total_cost,trans_date,trans_time,week_day,payment_method)
27      mycursor.execute(sqlFormula)
28      mydb.commit() #Save Data
29      card_id = np.random.choice(cardAll)
30      store_id = np.random.choice(storeAll)
31      randomMy = np.random.choice(range(4500))
32      for j in range(np.random.choice(range(100))):
33          barcode = ab['barcode'][j + randomMy + 1]
34          quantity = np.random.choice(range(1,6))
35          sqlFormula = """INSERT INTO bought
            ↪  (transact_id,card_id,store_id,quantity,barcode)
36                      VALUES
    ↪  ({},{},{},{},{})""".format(i+1,card_id,store_id,quantity,barcode)
37          mycursor.execute(sqlFormula)
38      mydb.commit()
```

4.2 BackEnd Server

```
1  import os
2  from flask import Flask, render_template, request
3  from flask_mysqldb 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"
```

```
mysql = MySQL(app)


def shutdown_server():
    func = request.environ.get("werkzeug.server.shutdown")
    if func is None:
        raise RuntimeError("Not running with the Werkzeug Server")
    func()


@app.route("/")
def mainIndex():
    return render_template("layout.html")


@app.route("/product_bought_together", methods=["GET", "POST"])
def product_bought_together():
    cur = mysql.connection.cursor()
    my_query = """SELECT
    A.barcode, B.barcode, COUNT(A.barcode)
FROM
    bought A,
    bought B
WHERE
    A.transact_id = B.transact_id
        AND A.barcode > B.barcode
GROUP BY A.barcode , B.barcode
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 100;
"""
    print(my_query)
    cur.execute(my_query)
    results = cur.fetchall()
    return render_template("music3.html", results=results)


@app.route("/popular_positions", methods=["GET", "POST"])
def popular_positions():
    cur = mysql.connection.cursor()
    my_query = """SELECT
    A.store_id, A.shelf, A.alley, COUNT(B.barcode)
FROM
    has A,
    bought B
WHERE
    A.barcode = B.barcode
GROUP BY B.barcode , A.store_id , A.shelf , A.alley
ORDER BY COUNT(B.barcode) DESC
```

```
LIMIT 100;
"""
    print(my_query)
    cur.execute(my_query)
    results = cur.fetchall()
    return render_template("music4.html", results=results)


@app.route("/label_trust", methods=["GET", "POST"])
def label_trust():
    cur = mysql.connection.cursor()
    my_query = """SELECT
   A.category,
   COUNT(B.barcode) / E.total * 100 AS percentage
FROM
   product A,
   bought B,
   (SELECT COUNT(C.barcode) as total, D.category as categor
   FROM bought C, product D
   WHERE C.barcode = D.barcode
   GROUP BY D.category)E
WHERE
    A.barcode = B.barcode
   and A.category = E.categor
   AND A.special_note = 1
GROUP BY A.special_note , A.category
ORDER BY percentage DESC;
"""
    print(my_query)
    cur.execute(my_query)
    results = cur.fetchall()
    return render_template("music5.html", results=results)


@app.route("/hours_spent_more", methods=["GET", "POST"])
def hours_spent_more():
    cur = mysql.connection.cursor()
    my_query = """SELECT
    HOUR(trans_time),
    CAST(SUM(A.total_cost) AS DECIMAL (20 , 2 ))
FROM
    transact A,
    bought B
WHERE
    A.transact_id = B.transact_id
GROUP BY HOUR(trans_time)
ORDER BY SUM(A.total_cost) DESC , HOUR(trans_time)
"""
```

```
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
    ↪ 20',
118     cast(SUM(IF(A.age BETWEEN 20 AND 29, 1, 0))/totals.total * 100  as
    ↪ decimal(20,2)) AS '20 - 29',
119     cast(SUM(IF(A.age BETWEEN 30 AND 39, 1, 0))/totals.total * 100 as
    ↪ decimal(20,2)) AS '30 - 39',
120     cast(SUM(IF(A.age BETWEEN 40 AND 49, 1, 0))/totals.total * 100 as
    ↪ decimal(20,2)) AS '40 - 49',
121     cast(SUM(IF(A.age BETWEEN 50 AND 59, 1, 0))/totals.total * 100 as
    ↪ decimal(20,2)) AS '50 - 59',
122     cast(SUM(IF(A.age BETWEEN 60 AND 69, 1, 0))/totals.total * 100 as
    ↪ decimal(20,2)) AS '60 - 69',
123     cast(SUM(IF(A.age >= 70, 1, 0))/totals.total * 100 as decimal (20,2)) AS 'Over
    ↪ 70',
124     HOUR(C.trans_time) AS Hour
125 FROM
126     (SELECT distinct
127         YEAR(CURDATE()) - YEAR(birth_date) -
    ↪ IF(STR_TO_DATE(CONCAT(YEAR(CURDATE()), '-', MONTH(birth_date), '-',
    ↪ DAY(birth_date)), '%Y-%c-%e') > CURDATE(), 1, 0) AS age,
128             card_id
129     FROM
130         customer) A,
131     (SELECT * from bought group by transact_id) B
132     LEFT JOIN
133     transact C on B.transact_id = C.transact_id,
134     (SELECT distinct
135         COUNT(D.transact_id) as total, HOUR(D.trans_time) as total_hour
136         FROM
137             transact D
138         GROUP BY HOUR(D.trans_time)
139         ORDER BY HOUR(D.trans_time)) totals
140 WHERE
141     A.card_id = B.card_id
142         AND B.transact_id = C.transact_id
143         AND totals.total_hour = HOUR(C.trans_time)
144 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),
    ↪  A.store_id
301 FROM
302     bought A,
303     bought B,
304     has C,
305     has D
306 WHERE
307     A.transact_id = B.transact_id
308         AND A.barcode > B.barcode
309         AND A.barcode = C.barcode
310         AND A.store_id = C.store_id
311         AND B.barcode = D.barcode
312         AND B.store_id = 1
313         AND B.store_id = D.store_id
314 GROUP BY A.barcode , B.barcode, A.store_id
315 ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
316 LIMIT 1)
317 union
318 (SELECT
319     A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
    ↪  A.store_id
320 FROM
321     bought A,
322     bought B,
323     has C,
324     has D
325 WHERE
326     A.transact_id = B.transact_id
327         AND A.barcode > B.barcode
328         AND A.barcode = C.barcode
```

```
        AND A.store_id = C.store_id
        AND B.barcode = D.barcode
        AND B.store_id = 2
        AND B.store_id = D.store_id
GROUP BY A.barcode , B.barcode, A.store_id
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 1)
union
(SELECT
    A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
↪   A.store_id
FROM
    bought A,
    bought B,
    has C,
    has D
WHERE
    A.transact_id = B.transact_id
        AND A.barcode > B.barcode
        AND A.barcode = C.barcode
        AND A.store_id = C.store_id
        AND B.barcode = D.barcode
        AND B.store_id = 3
        AND B.store_id = D.store_id
GROUP BY A.barcode , B.barcode, A.store_id
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 1)
union
(SELECT
    A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
↪   A.store_id
FROM
    bought A,
    bought B,
    has C,
    has D
WHERE
    A.transact_id = B.transact_id
        AND A.barcode > B.barcode
        AND A.barcode = C.barcode
        AND A.store_id = C.store_id
        AND B.barcode = D.barcode
        AND B.store_id = 4
        AND B.store_id = D.store_id
GROUP BY A.barcode , B.barcode, A.store_id
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 1)
union
```

```
(SELECT
    A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
↪   A.store_id
FROM
    bought A,
    bought B,
    has C,
    has D
WHERE
    A.transact_id = B.transact_id
        AND A.barcode > B.barcode
        AND A.barcode = C.barcode
        AND A.store_id = C.store_id
        AND B.barcode = D.barcode
        AND B.store_id = 5
        AND B.store_id = D.store_id
GROUP BY A.barcode , B.barcode, A.store_id
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 1)
union
(SELECT
    A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
↪   A.store_id
FROM
    bought A,
    bought B,
    has C,
    has D
WHERE
    A.transact_id = B.transact_id
        AND A.barcode > B.barcode
        AND A.barcode = C.barcode
        AND A.store_id = C.store_id
        AND B.barcode = D.barcode
        AND B.store_id = 6
        AND B.store_id = D.store_id
GROUP BY A.barcode , B.barcode, A.store_id
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 1)
union
(SELECT
    A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
↪   A.store_id
FROM
    bought A,
    bought B,
    has C,
    has D
```

```
WHERE
    A.transact_id = B.transact_id
        AND A.barcode > B.barcode
        AND A.barcode = C.barcode
        AND A.store_id = C.store_id
        AND B.barcode = D.barcode
        AND B.store_id = 7
        AND B.store_id = D.store_id
GROUP BY A.barcode , B.barcode, A.store_id
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 1)
union
(SELECT
    A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
    A.store_id
FROM
    bought A,
    bought B,
    has C,
    has D
WHERE
    A.transact_id = B.transact_id
        AND A.barcode > B.barcode
        AND A.barcode = C.barcode
        AND A.store_id = C.store_id
        AND B.barcode = D.barcode
        AND B.store_id = 8
        AND B.store_id = D.store_id
GROUP BY A.barcode , B.barcode, A.store_id
ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
LIMIT 1)
union
(SELECT
    A.barcode,C.shelf,C.alley, B.barcode,D.shelf,D.alley, COUNT(A.barcode),
    A.store_id
FROM
    bought A,
    bought B,
    has C,
    has D
WHERE
    A.transact_id = B.transact_id
        AND A.barcode > B.barcode
        AND A.barcode = C.barcode
        AND A.store_id = C.store_id
        AND B.barcode = D.barcode
        AND B.store_id = 9
        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),
     ↪  A.store_id
472  FROM
473      bought A,
474      bought B,
475      has C,
476      has D
477  WHERE
478      A.transact_id = B.transact_id
479          AND A.barcode > B.barcode
480          AND A.barcode = C.barcode
481          AND A.store_id = C.store_id
482          AND B.barcode = D.barcode
483          AND B.store_id = 10
484          AND B.store_id = D.store_id
485  GROUP BY A.barcode , B.barcode, A.store_id
486  ORDER BY COUNT(A.barcode) DESC , A.barcode , B.barcode
487  LIMIT 1)
488      """
489      print(my_query)
490      cur.execute(my_query)
491      results = cur.fetchall()
492      return render_template("query_2.html", results=results)
493
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 editProductsResponce():
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,
```

```
            price,
            special_note,
            category,
        ]
        additionalQuery = list(filter(lambda i: i != "", additionalQuery))
        if additionalQuery != []:
            attributes, values = zip(*additionalQuery)
            attributes = ",".join(attributes)
            values = ",".join(values)
            my_query = (
                "insert into product (" + attributes + ") values (" + values + ")"
            )
            print(my_query)
            cur = mysql.connection.cursor()
            cur.execute(my_query)
            mysql.connection.commit()

    cur = mysql.connection.cursor()
    my_query = "select * from product"
    cur.execute(my_query)
    results = cur.fetchall()
    return render_template("editProductsResponce.html", albums=results)


@app.route("/editCustomers")
def editCustomers():
    return render_template("editCustomers.html")


@app.route("/search_resultC", methods=["GET", "POST"])
def editCustomersResponce():
    my_data = request.get_json(force=True)
    edit = int(my_data.get("edit"))
    print(edit)
    if edit == -1:
        card_id = my_data.get("card_id")
        if card_id != "":
            cur = mysql.connection.cursor()
            my_query = "DELETE FROM customer WHERE card_id = {}".format(card_id)
            cur.execute(my_query)
            mysql.connection.commit()
    elif edit == 0:
        card_id = my_data.get("card_id")
        if card_id != "":
            # Update ...

            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 != "",
                ↪  additionalQuery)))
753             if additionalQuery != "":
754                 my_query = (
755                     "update customer set "
756                     + additionalQuery
757                     + " where card_id = {};".format(my_data.get("card_id"))
758                 )
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 != "")
            ↪  else ""
782
783         data = my_data.get("social_security_number")
784         social_security_number = (
785             ("social_security_number", "'" + data + "'") if (data != "") else ""
786         )
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             print(my_query)
806             cur = mysql.connection.cursor()
807             cur.execute(my_query)
808             mysql.connection.commit()
809 
810             phone_number = my_data.get("phone_number")
811             if phone_number != "":
812                 editNumber = int(my_data.get("editNumber"))
813                 phone_number = phone_number.split(",")
814                 if editNumber == 1:
815                     card = "select max(card_id) from customer;"
816                     cur = mysql.connection.cursor()
817                     cur.execute(card)
818                     card_id = cur.fetchall()[0][0]
819                     mysql.connection.commit()
820                     for phone in phone_number:
821                         my_query = "insert into customer_phone
                            ↪  (card_id,phone_number) values ('{}','{}');".format(
822                             card_id, phone
823                         )
824                         print(my_query)
825                         cur = mysql.connection.cursor()
826                         cur.execute(my_query)
827                         mysql.connection.commit()
828 
829     cur = mysql.connection.cursor()
830     my_query = "select A.*, B.phone_number from customer A, customer_phone B where
        ↪  B.card_id = A.card_id;"
831     cur.execute(my_query)
832     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
            ↪  year(transact.trans_date) = year('{0}')".format(
862             my_data.get("birthday")
863         )
864         if (my_data.get("birthday") != "")
865         else ""
866     )
867     quantity = (
868         " and bought.quantity >= {}".format(my_data.get("quantity"))
869         if (my_data.get("quantity") != "")
870         else ""
871     )
872     barcode = (
873         " and bought.barcode = {}".format(my_data.get("barcode"))
874         if (my_data.get("barcode") != "")
875         else ""
876     )
877     total_cost = (
878         " and transact.total_cost >= {}".format(my_data.get("total_amount"))
879         if (my_data.get("total_amount") != "")
```

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

```
                from transact
                where transact_id in (
                 select transact_id
                 from bought
                  where card_id = {}
                )
                group by hour(trans_time)
                order by hour(trans_time)""".format(
        actual_data
    )
    cur.execute(myquery4)
    result = []
    x = ("x", "y")
    for row in cur:
        result.append(dict(zip(x, row)))
    print(result)
    return render_template("customers.html", result=result)


@app.route("/customers_visit_data")
def customers():
    return render_template("customers_visit_data.html")


@app.route("/customer_result", methods=["GET", "POST"])
def customer_result():
    my_data = request.get_json(force=True)
    my_useful_data = my_data["card_id"]
    cur = mysql.connection.cursor()
    my_query_2 = "SELECT distinct b.card_id, b.store_id FROM bought as b, store as
    ↪  s where b.store_id = s.store_id AND b.card_id = {} order by card_id,
    ↪  store_id".format(
        my_useful_data
    )
    print(my_query_2)
    cur.execute(my_query_2)
    results2 = cur.fetchall()
    my_query_3 = """
        SELECT card_id, barcode, sum(quantity)
        FROM bought
        where card_id = {}
        group by barcode
        order by sum(quantity) desc
        limit 10
        """.format(
        my_useful_data
    )
    print(my_query_3)
```

```
    cur.execute(my_query_3)
    results3 = cur.fetchall()
    my_query_4 = """
    SELECT
    b.card_id, MONTHNAME(t.trans_date), cast(AVG(t.total_cost) as decimal(6,2))
FROM
    transact AS t,
    bought AS b
WHERE
    b.card_id = {}
   AND b.transact_id = t.transact_id
GROUP BY MONTHNAME(t.trans_date)
ORDER BY MONTH(t.trans_date);
""".format(
        my_useful_data
    )
    print(my_query_4)
    cur.execute(my_query_4)
    results4 = cur.fetchall()
    my_query_5 = """
    SELECT
 b.card_id,
    WEEK(t.trans_date) Week,
    cast(AVG(total_cost) as decimal(6,2))
FROM
    bought AS b,
    transact AS t
WHERE
 b.card_id = {}
    AND b.transact_id = t.transact_id
GROUP BY WEEK(t.trans_date)
""".format(
        my_useful_data
    )
    print(my_query_5)
    cur.execute(my_query_5)
    results5 = cur.fetchall()

    # the_id = request.args.get('button_id')
    return render_template(
        "music2.html",
        results=results2,
        results3=results3,
        results4=results4,
        results5=results5,
    )


```

```python
@app.route("/report2", methods=["GET", "POST"])
def report2():
    abduction = {
        "firstname": request.form["firstname"],
        "lastname": request.form["lastname"],
    }
    return render_template("report2.html", abduction=abduction)


# start the server
if __name__ == "__main__":
    app.run(
        host=os.getenv("IP", "0.0.0.0"), port=int(os.getenv("PORT", 8587)),
        ↪  debug=True
    )
```

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