BITS & BOTS PROJECT

The Ohio State University

CSE 3241 AU20

Team 6

Donghyun Kim

Quynh Bui

Hung-Hsiu Yen

Cade Sbrocco

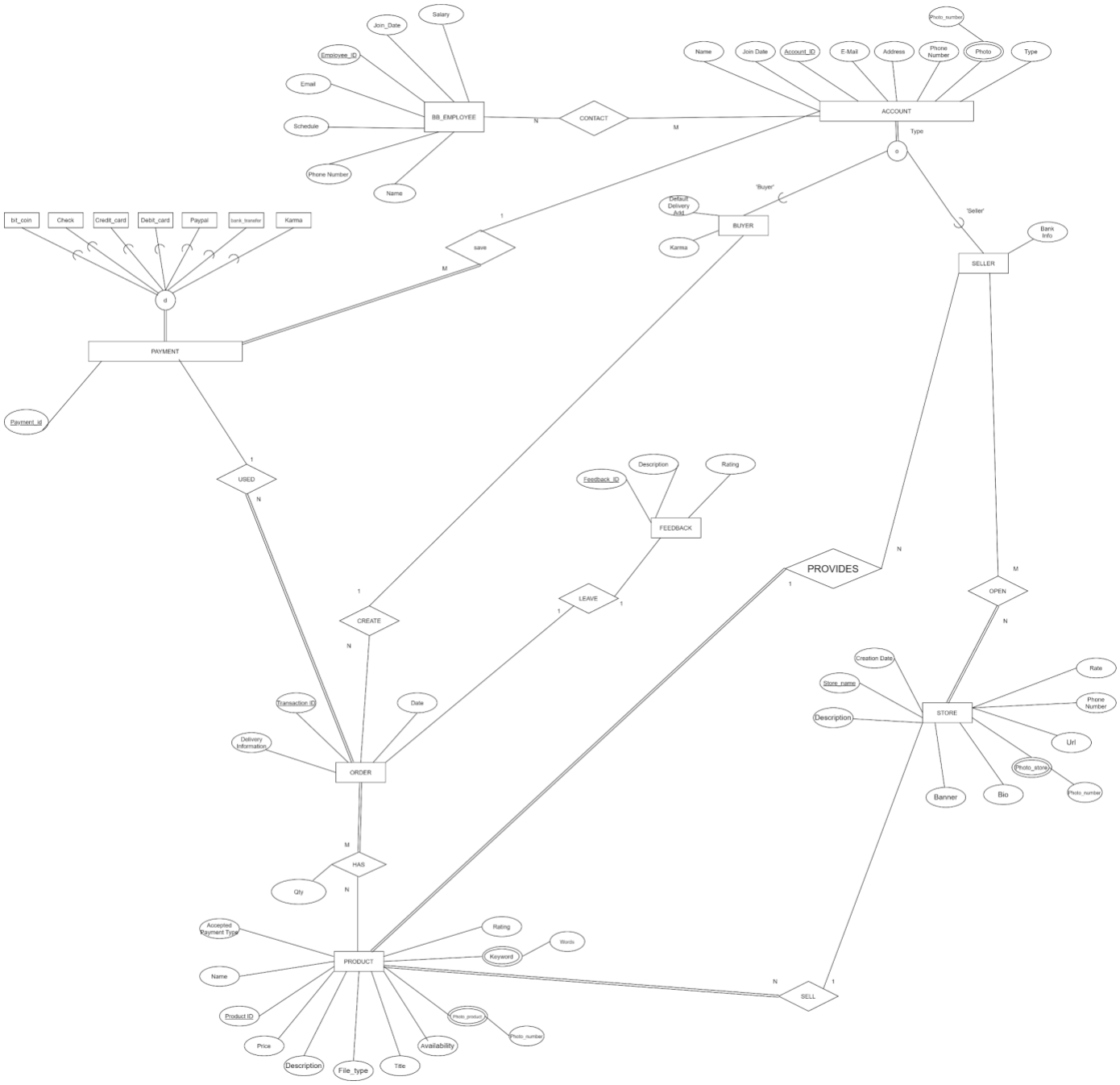
All team members contributed equally to all parts of this project.

TABLE OF CONTENT

1. The Final Report

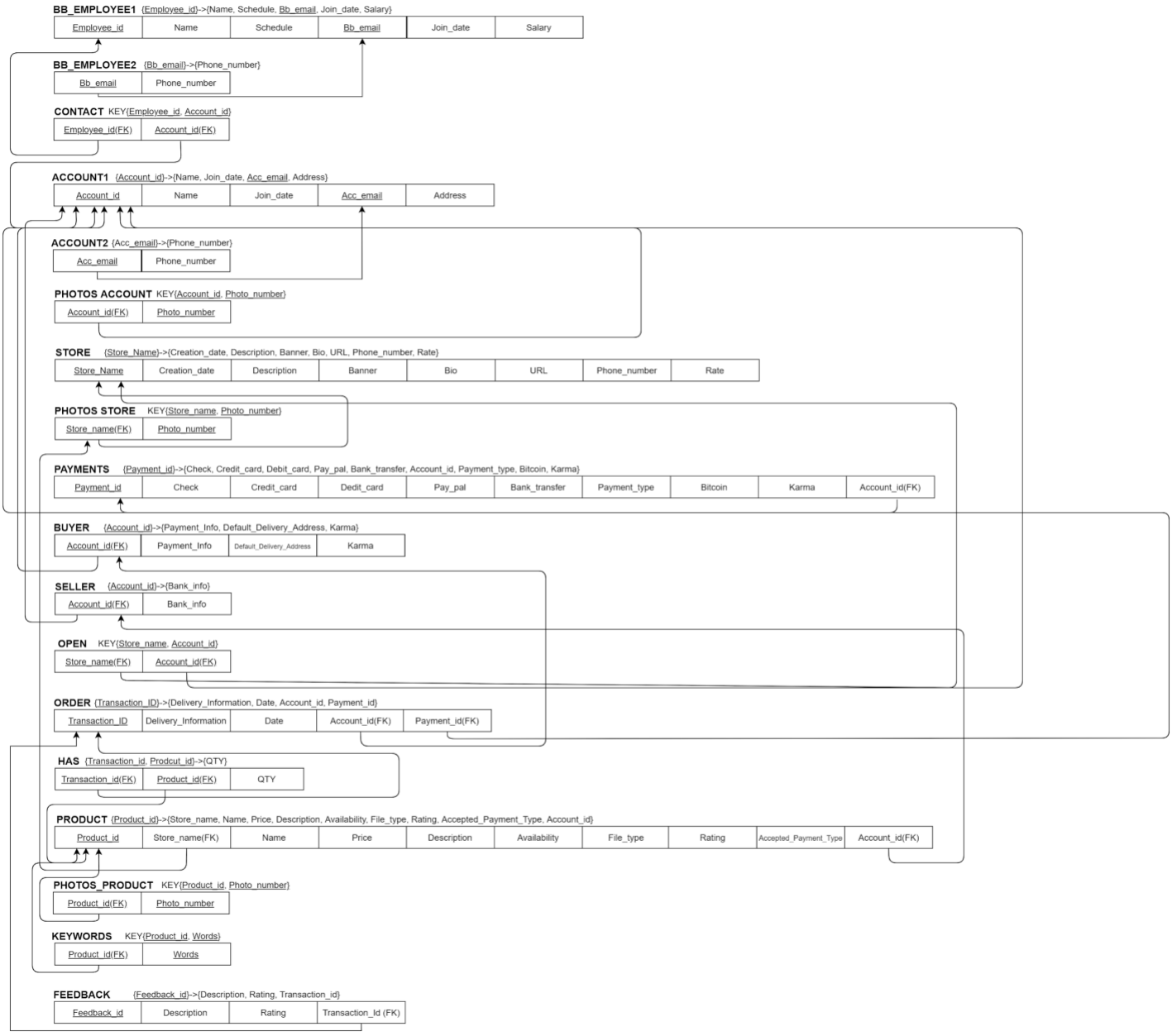
1.1 Database Description

1.1.1 ER-Model



Entities	Attributes (and other info)
ACCOUNT	Subclasses: BUYER, SELLER. Attributes: **Account_id**, name, **e-mail**, address, phone_number, photo (multivalued), join_date, type
BUYER	Attributes: default_payment_information, Default_delivery_information, karma
SELLER	Attributes: karma_value, bank_information
ORDER	Attributes: **Transaction_ID**, date, payment_information (multivalued), delivery_information
PRODUCT	Attributes: Price, name, availability, keyword (multivalued), **product_ID,** title, description, image (multivalued), file_type, type of payment accepted, rating
FEEDBACK	Attributes: rating, description, **feedback_id**
STORE	Attribute: **Name**, description, banner, bio, photos, url, phone number, creation date
BB_EMPLOYEE	Attribute: **Employee_ID**, e-mail, phone number, name, salary, schedule, join_date
PAYMENT	Attribute: **Payment_id,** Check, Credit card, Debit card, Paypal, Bank transfer, karma, bit_coin

1.1.2 Relational Schema



1.1.3 Database Normalization

All the tables are in BCNF, because from our functional dependencies above, attributes in the left is the superkey that determines the uniqueness of the tuples.

1.1.4 Relational Schema and SQL for 2 views

- List all the products, and the total of each sold in descending order. This view is helpful in seeing the top selling products on the site.

```
CREATE VIEW PRODUCT_QTY AS
SELECT      P.Product_id, SUM(H.QTY)
FROM        (PRODUCT AS P JOIN HAS AS H ON
P.Product_id=H.Product_id)
GROUP BY    P.Product_id
ORDER BY    SUM(H.QTY) DESC;
```

Relational algebra expression:

$PRODUCT\_QTY \leftarrow \pi_{Product\_id,\ Sum\_Qty}(PRODUCT \bowtie_{Product\_id=Product\_id} ({}_{Product\_id}\mathcal{F}_{SUM\ QTY}(HAS)))$

Sample output:

Product_id	SUM(H.QTY)
P0002	130
P0007	87
P0009	67
P0010	65
P0006	55
P0003	53
P0012	49
P0005	42
P0013	34
P0008	33
P0001	32
P0011	30
P0020	24
P0019	2

- List all the sellers and the number of unique products that they sell in descending order. This view is helpful to see the sellers with the widest variety of products.

```
CREATE VIEW      SELLER_PRODUCT   AS
SELECT      S.Account_id, COUNT(P.Product_id)
FROM        (SELLER AS S JOIN PRODUCT AS P ON S.Account_id =
P.Account_id)
GROUP BY    S.Account_id
ORDER BY    COUNT(P.Product_id) DESC;
```

Relational algebra expression:

$SELLER\_PRODUCT \leftarrow \pi_{Account\_id,\ Count\_Qty}(SELLER \bowtie_{Account\_id=Accountt\_id} ({}_{Product\_id}\mathcal{F}_{Count\ QTY}(PRODUCT)))$

Sample output:

Account_id	COUNT(P.Product_id)
S0009	3
S0002	3
S0001	3
S0010	2
S0007	2
S0005	2
S0004	2
S0003	2
S0006	1

1.1.5 Indexes

Hash: For hash index, we choose the words attribute from the KEYWORD schema since this attribute will be used when customers want to search the item by keywords. Which will be an equality test.

B-tree: We would index the price attribute of a PRODUCT as a B-tree since this attribute will be used in many range tests. For example, if a user wants to search for products within a certain price range.

Below is the SQL code to create the two indices above:

```
CREATE INDEX price_index
ON PRODUCT(price);

CREATE INDEX keyword_index
ON KEYWORDS(words);
```

1.1.6 Transactions

- Make a new order. This creates an order and decreases the quantity of the product sold. In addition, if the user attempts to buy more than available, the transaction will not be filled.

```
BEGIN TRANSACTION;
      INSERT OR ROLLBACK INTO ORDER VALUES ('T0021', NULL, '2020-12-
      01', 'B0007',17);
      INSERT OR ROLLBACK INTO HAS VALUES ('T0021', 'P0008', 20);
      UPDATE OR ROLLBACK PRODUCT
            SET Availability = Availability-20
            WHERE Product_id = 'T0021';
 COMMIT;
```

- Open a new store. This creates not only the store but adds pictures to the store when it is created.

```
BEGIN TRANSACTION;
      INSERT OR ROLLBACK INTO STORE VALUES ('ABC Store', '2020-02-
14', NULL, NULL, NULL, NULL, 2163526351, NULL);
      INSERT OR ROLLBACK INTO PHOTOS_STORE ('ABC Store', 321);
      INSERT OR ROLLBACK INTO OPEN VALUES ('ABC Store','S0009');
COMMIT;
```

1.2 User Manual

1.2.1 Description of Extra Entities

Entity	Read World	Attributes (and other info)
STORE	Sellers can open stores in our system.	Attribute: **Name(string)**, description(string), banner(string), bio(string), photos(int), url(string), phone number(int), creation date(date)
BB_EMPLOYEE	Employees who work in our system, and provide customer service to the users.	Attribute: **Employee_ID(string)**, e-mail(string), phone number(int), name(string), salary(int), schedule(string), join_date(data)
PAYMENT	Payment methods.	Attribute: **Payment_id(string),** Check(int), Credit card(int), Debit card(int), Paypal(int), Bank transfer(int), karma(int), bit_coin(int), payment_type(string)

1.2.2 Sample SQL Queries

(1) Find the buyer who has purchased the most IP Items and the total number of IP Items they have purchased

```
SELECT A.Name,    COUNT(*)
FROM ACCOUNT1 AS A, ORDER_ AS ORD, HAS AS HA, PRODUCT AS PR
WHERE A.Account_id = ORD.Account_id AND ORD.Transaction_id =
HA.Transaction_id AND HA.Product_id = PR.Product_id
GROUP BY A.Name
HAVING COUNT(*)=
      (SELECT MAX(BC)
      FROM (SELECT   COUNT(*) AS BC
            FROM ACCOUNT1 AS A, ORDER_ AS ORD, HAS AS HA, PRODUCT AS PR
```

```
                WHERE A.Account_id = ORD.Account_id AND
        ORD.Transaction_id = HA.Transaction_id AND HA.Product_id =
        PR.Product_id
              GROUP BY A.Name) );
```

(2) Give all the buyers who purchased a IP Item by a given seller and the names of the IP Items they purchased, seller given by Account_id = 'S0003'

```
SELECT b.*, p.Name
FROM PRODUCT AS p, BUYER AS b, ORDER_ AS o, HAS AS h, ACCOUNT1 AS a
WHERE p.Account_id = 'S0003' AND  h.Transaction_id = o.Transaction_id
AND o.Account_id = 'B0001' AND h.Product_id = p.Product_id AND
b.Account_id = a.Account_id;
```

1.2.3 INSERT Samples

(1) Insert items into product

```
INSERT INTO PRODUCT
VALUES
('P0021', 'Go Shop', 'jrr3',20, NULL, 200, 'txt', 4, NULL, 'S0001'),
('P0022', 'Nava', 'repo',4, NULL, 500, 'txt', 2, NULL, 'S0004'),
('P0023', 'Luxx', 'play',10, NULL, 100, 'txt', 4, NULL, 'S0002'),
('P0024', 'Go Shop','stop here',40, NULL, 800, 'txt', 5, NULL,
'S0001'),
('P0025', 'Ama Accessory', 'vi',12, NULL, 500, 'txt', 3, NULL,
'S0005');
```

(2) Insert new stores into store

```
INSERT INTO STORE
VALUES
('Gold', '2008-01-20', NULL, NULL, NULL, NULL, 6142738273, 4),
('Luxy', '2015-05-23', NULL,NULL,NULL,NULL, 6142842842, 5),
('Naria', '2014-09-20', NULL,NULL,NULL,NULL, 6141192288, 3),
('Hyper', '2022-05-04', NULL, NULL, NULL,NULL, 6142902008, 4),
('Ale Accessory, '2012-12-08', NULL, NULL, NULL,NULL, 6147262008, 5),
('Happy Dive', '2013-12-12', NULL, NULL, NULL, NULL, 6148301820, 5);
```

For more, see appendix.

1.2.4 DELETE Samples

(1)Delete a store name called Go Shop

```
DELETE FROM STORE
WHERE Store_name = 'Go Shop';
```

(2)Delete a product “P0001” from PRODUCT

```
DELETE FROM PRODUCT
WHERE Product_id = 'P0001';
```

Discussion:

- Deleting a buyer account cascades delete into their payments, orders, contact with employees, and feedback left.
- Deleting a seller account cascades delete into their stores, products, feedback received.
- Deleting an account of any kind cascades delete into their photos and their email/phone information.
- Deleting an order cascades delete to the tables that hold the quantity of each product in the order.
- Deleting a product cascades delete to any feedback received on the product and any orders containing the product.

1.3 Graded Checkpoint Documents

1.3.1 Original Checkpoints

See original checkpoints at appendix

1.3.2 Checkpoint Feedbacks and Revisions

Checkpoint	Feedbacks	Revisions
CP01	Each account should have multiple payment methods. Nice job overall.	Added payment methods (Debit card, Credit card, Paypal, Check, Bank Transfer)

CP02	Payment should be an entity instead of the attribute, and then do not need payment info attributes in other entities	Added PAYMENT entity with Debit card, Credit card, Paypal, Check, Bank Transfer as attributes. Later further modified to have payment superclass, with payment types being specializations. See entity diagram for updated view.
CP03	3.c syntax errors in having statement; 4.b syntax errors; 4.c columns does not exist; 5.a columns does not exist; 5.b: columns miss match; missing 5.e - 5.g	Fixed and created as needed. Can be seen in the queries section of report.
CP04	In BB_EMPLOYEE, email should be a candidate key so that can determine others, need to break down the schema; same for account; In order, transaction_id should be able to determine the rests, renormalize the schema accordingly; missing error checking in transactions	- Replaced BB_EMPLOYEE by BB_EMPLOYEE1 and BB_EMPLOYEE2, ACCOUNT by ACCOUNT1 and ACCOUNT2. This can be seen in our diagram, schema, and insert sections - Fixed transactions, see transaction section of report.

2. The SQL Database

2.1 Database Description

2.1.1 SQL CREATE

a. Create a BB_EMPLOYEE1 table

```
CREATE TABLE BB_EMPLOYEE1
(Employee_id          VARCHAR(15)      NOT NULL,
 Name       VARCHAR(15)      NOT NULL,
Schedule   VARCHAR(15)      NOT NULL,
Email            VARCHAR(100)     NOT NULL,
Join_date        DATE       NOT NULL,
Salary                INT   NOT NULL,
PRIMARY KEY(Employee_id    ));
```

b. Create a BB_EMPLOYEE2 table

```
CREATE TABLE BB_EMPLOYEE2
(Email           VARCHAR(100)     NOT NULL,
Phone_number     INT        ,
FOREIGN KEY (Email) REFERENCES BB_EMPLOYEE1(Email)
ON DELETE SET NULL    ON UPDATE CASCADE);
```

c. Create a CONTACT table

```
CREATE TABLE CONTACT
(Employee_id          VARCHAR(15)      NOT NULL,
Account_id       VARCHAR(15)      NOT NULL,
FOREIGN KEY(Employee_id    ) REFERENCES BB_EMPLOYEE1(Employee_id),
FOREIGN KEY(Account_id) REFERENCES ACCOUNT1(Account_id));
```

d. Create an ACCOUNT1 table

```
CREATE TABLE ACCOUNT1
(Account_id           VARCHAR(15)      NOT NULL,
 Name            VARCHAR(15)      NOT NULL,
Join_date             DATE       NOT NULL,
Email                 VARCHAR(100)     NOT NULL,
Address          VARCHAR(100)     ,
PRIMARY KEY(Account_id));
```

e. Create an ACCOUNT2 table

```
CREATE TABLE ACCOUNT2
(Email                VARCHAR(100)     NOT NULL,
Phone_number     INT        ,
FOREIGN KEY(Email) REFERENCES ACCOUNT1(Email)
```

```
ON DELETE CASCADE     ON UPDATE CASCADE);
```

f. Create a PHOTOS ACCOUNT table

```
CREATE TABLE PHOTOS_ACCOUNT
(Account_id           VARCHAR(15)      NOT NULL,
Photo_number     INT   NOT NULL,
PRIMARY KEY(Account_id, Photo_number),
FOREIGN KEY(Account_id) REFERENCES ACCOUNT1(Account_id)
ON DELETE CASCADE     ON UPDATE CASCADE);
```

g. Create a STORE table

```
CREATE TABLE STORE
(Store_name           VARCHAR(30)      NOT NULL,
Creation_date         DATE       NOT NULL,
Description           VARCHAR(100),
Banner                VARCHAR(30),
Bio              VARCHAR(100),
URL              VARCHAR(50),
Phone            INT,
Rate             INT,
PRIMARY KEY(Store_name));
```

h. Create a PHOTOS_STORE table

```
CREATE TABLE PHOTOS_STORE
(Store_name           VARCHAR(30)      NOT NULL,
 Photo_number    INT              NOT NULL,
PRIMARY KEY (Store_name, Photo_number),
FOREIGN KEY(Store_name) REFERENCES STORE(Store_name)
ON DELETE CASCADE     ON UPDATE CASCADE);
```

i. Create a PAYMENTS table

```
CREATE TABLE PAYMENTS
(Payment_id           INT   NOT NULL,
Bank_check       INT,
Credit_card           INT,
Debit_card       INT,
Paypal                INT,
Bank_transfer         INT,
Payment_type          VARCHAR(15),
Bitcoin               INT,
Karma            INT,
Account_id       VARCHAR(15)      NOT NULL,
PRIMARY KEY (Payment_id),
FOREIGN KEY (Account_id) REFERENCES ACCOUNT1(Account_id)
ON DELETE CASCADE     ON UPDATE CASCADE);
```

j. Create a BUYER table

```
CREATE TABLE BUYER
(Account_id           VARCHAR(15)           NOT NULL,
Payment_info          VARCHAR(100),
Default_Delivery_Address         VARCHAR(100),
Karma      INT
     CHECK(KARMA>-1),
FOREIGN KEY (Account_id) REFERENCES ACCOUNT1(Account_id)
ON DELETE CASCADE     ON UPDATE CASCADE);
```

k. Create a SELLER table

```
CREATE TABLE SELLER
(Account_id           VARCHAR(15)           NOT NULL,
Bank_info        VARCHAR(100),
FOREIGN KEY (Account_id) REFERENCES ACCOUNT1(Account_id)
ON DELETE CASCADE     ON UPDATE CASCADE);
```

l. Create an OPEN table

```
CREATE TABLE OPEN
(Store_name           VARCHAR(30)      NOT NULL,
Account_id       VARCHAR(15)      NOT NULL,
FOREIGN KEY(Store_name)  REFERENCES STORE(Store_name)
ON DELETE CASCADE     ON UPDATE CASCADE,
FOREIGN KEY(Account_id)    REFERENCES ACCOUNT1(Account_id)
ON DELETE CASCADE     ON UPDATE CASCADE);
```

m. Create an ORDER_ table

```
CREATE TABLE ORDER_
(Transaction_id  VARCHAR(20)      NOT NULL,
Delivery_info         VARCHAR(20),
Date_of_order              DATE,
Account_id            VARCHAR(15)      NOT NULL,
Payment_id       VARCHAR(15)      NOT NULL,
PRIMARY KEY(Transaction_id),
FOREIGN KEY(Account_id)  REFERENCES ACCOUNT1(Account_id)
ON DELETE CASCADE     ON UPDATE CASCADE,
FOREIGN KEY(Payment_id)  REFERENCES PAYMENTS(Payment_id)
ON DELETE CASCADE     ON UPDATE CASCADE);
```

n. Create a HAS table

```
CREATE TABLE HAS
(Transaction_id  VARCHAR(20)      NOT NULL,
Product_id       VARCHAR(20)      NOT NULL,
Qty              INT              NOT NULL
```

```
     CHECK(Qty >0),
FOREIGN KEY(Transaction_id)  REFERENCES ORDER_(Transaction_id)
ON DELETE CASCADE     ON UPDATE CASCADE,
FOREIGN KEY(Product_id)  REFERENCES PRODUCT(Product_id)
ON DELETE CASCADE     ON UPDATE CASCADE);
```

o. Create a PRODUCT table

```
CREATE TABLE PRODUCT
(Product_id           VARCHAR(20)      NOT NULL,
Store_name       VARCHAR(30)      NOT NULL,
Name             VARCHAR(20)      NOT NULL,
Price            INT
           CHECK(Price > 0),
Description           VARCHAR(100),
Availability          INT
           CHECK(Availability>-1),
File_type        VARCHAR(20),
Rating                INT,
Accepted_payment_type VARCHAR(15)      NOT NULL,
Account_id            VARCHAR(15)      NOT NULL,
PRIMARY KEY(Product_id),
FOREIGN KEY(Store_name)  REFERENCES STORE(Store_name)
ON DELETE CASCADE     ON UPDATE CASCADE,
FOREIGN KEY(Account_id)  REFERENCES ACCOUNT1(Account_id)
ON DELETE CASCADE     ON UPDATE CASCADE);
```

p. Create a PHOTOS_PRODUCT table

```
CREATE TABLE PHOTOS_PRODUCT
(Product_id           VARCHAR(10)      NOT NULL,
 Photo_number    INT              NOT NULL,
PRIMARY KEY (Product_id, Photo_number),
FOREIGN KEY (Product_id) REFERENCES PRODUCT(Product_id)
     ON DELETE CASCADE     ON UPDATE CASCADE);
```

q. Create a KEYWORDS table

```
CREATE TABLE KEYWORDS
(Product_id           VARCHAR(10)      NOT NULL,
 Words                VARCHAR(20)      NOT NULL,
PRIMARY KEY (Product_id, Words),
FOREIGN KEY (Product_id) REFERENCES PRODUCT(Product_id)
     ON DELETE CASCADE     ON UPDATE CASCADE);
```

r. Create FEEDBACK table

```
CREATE TABLE FEEDBACK
(Feedback_id          VARCHAR(15)      NOT NULL,
```

```
 Description          VARCHAR(200)     ,
 Rating               INT              NOT NULL

           CHECK(Rating>-1 AND Rating <6),
Transaction_id        VARCHAR(15)      NOT NULL,

PRIMARY KEY (Feedback_id)
FOREIGN KEY (Transaction_id) REFERENCES ORDER_(Transaction_id)
     ON DELETE CASCADE     ON UPDATE CASCADE
);
```

s. Create index (price from PRODUCT)

```
CREATE INDEX price_index
ON PRODUCT(price);
```

t. Create index (word from KEYWORDS)

```
CREATE INDEX keyword_index
ON KEYWORDS(words);
```

u. Create a view: List all the products, and the total of each sold in descending order.

```
CREATE VIEW PRODUCT_QTY AS
SELECT     P.Product_id, SUM(H.QTY)
FROM       (PRODUCT AS P JOIN HAS AS H ON P.Product_id=H.Product_id)
GROUP BY   P.Product_id
ORDER BY   SUM(H.QTY) DESC;
```

v. Create a view: List all the sellers and the number of unique products that they sell in descending order

```
CREATE VIEW     SELLER_PRODUCT  AS
SELECT     S.Account_id, COUNT(P.Product_id)
FROM       (SELLER AS S JOIN PRODUCT AS P ON S.Account_id =
P.Account_id)
GROUP BY   S.Account_id
ORDER BY   COUNT(P.Product_id) DESC;
```

2.1.2 SQL INSERT

a. Insert sample of data to BB_EMPLOYEE1

```
INSERT INTO BB_EMPLOYEE1
VALUES
     ('E0001', 'Michael Scott',  'Schedule', 'scott1@gmail.com',
'2015-12-12', 70000),
     ('E0002', 'Dwight Schrute',  'Schedule', 'schrute09@gmail.com',
'2017-04-12', 60000),
```

```
      ('E0003', 'Jim Halpert',  'Schedule', 'halpert123@gmail.com',
'2017-06-01', 60000),
      ('E0004', 'Pam Beasley',  'Schedule', 'pampam@gmail.com',  '2017-
08-04', 50000),
      ('E0005', 'Andy Bernard', 'Schedule', 'bernard23@gmail.com',
'2018-01-12', 55000),
      ('E0006', 'Richard Korf', 'Schedule', 'korh13@gmail.com', '2018-
05-15', 85000),
      ('E0007', 'John Kim', 'Schedule', 'kim938@gmail.com', '2019-10-
18', 70000),
      ('E0008', 'Joseph Kay', 'Schedule', 'kay81@gmail.com', '2017-08-
03', 55000),
      ('E0009', 'Allen Klinger', 'Schedule', 'klinger05@gmail.com',
'2017-06-23', 90000),
      ('E0010', 'Paul Meyer', 'Schedule', 'meyer83@gmail.com', '2019-
01-29', 60000),
      ('E0011', 'James Johnson', 'Schedule', 'jj111@gmail.com', '2019-
01-23', 45000),
      ('E0012', 'Jake Jackson', 'Schedule', 'jj121@gmail.com', '2020-
02-21', 80000),
      ('E0013', 'Tom Smith', 'Schedule', 'ts33@gmail.com', '2017-05-
30', 55000),
      ('E0014', 'Greg Brown', 'Schedule', 'gb1256@gmail.com', '2019-09-
30', 70000),
      ('E0015', 'Tim Simpson', 'Schedule', 'ts6543@gmail.com', '2018-
04-23', 100000),
      ('E0016', 'Jerry Jackson', 'Schedule', 'jj123@gmail.com', '2020-
05-05', 50000),
      ('E0017', 'Jarrod Newsted', 'Schedule', 'jn123@gmail.com', '2016-
05-30', 51000),
      ('E0018', 'Jena Jackson', 'Schedule', 'jo113@gmail.com',  '2017-
05-05', 53000),
      ('E0019', 'James Hatfield', 'Schedule', 'js144@gmail.com',
'2019-05-05', 48000),
      ('E0020', 'Jerry Schemit', 'Schedule', 'js155@gmail.com', '2018-
05-05', 46000);
```

b. Insert sample of data to BB-EMPLOYEE2

```
INSERT INTO BB_EMPLOYEE2
VALUES
      ('scott1@gmail.com',  2162343212),
      ('schrute09@gmail.com', 6149003400),
      ('halpert123@gmail.com', 6145264993),
      ('pampam@gmail.com', 6145263934),
      ('bernard23@gmail.com', 6145279430),
```

```
      ('korh13@gmail.com', 6145264934),
      ('kim938@gmail.com', 6145628042),
      ('kay81@gmail.com', 7346351634),
      ('klinger05@gmail.com', 8273634931),
      ('meyer83@gmail.com', 7236238281),
      ('jj111@gmail.com', 3303743812),
      ('jj121@gmail.com', 6148284932),
      ('ts33@gmail.com', 6145262832),
      ('gb1256@gmail.com', 6145261232),
      ('ts6543@gmail.com', 2161827372),
      ('jj123@gmail.com', 4402832312),
      ('jn123@gmail.com', 2164826361),
      ('jo113@gmail.com', 6142536253),
      ('js144@gmail.com', 6147956666),
      ('js155@gmail.com', 6147977775);
```

c. Insert sample of data to CONTACT

```
INSERT INTO CONTACT
VALUES
      ('E0001', 'B0001'),
      ('E0002', 'B0002'),
      ('E0003', 'S0003'),
      ('E0004', 'B0004'),
      ('E0005', 'S0002'),
      ('E0006', 'B0006'),
      ('E0007', 'S0006'),
      ('E0008', 'B0007'),
      ('E0009', 'B0008'),
      ('E0010', 'B0010'),
      ('E0011', 'S0001'),
      ('E0012', 'S0002'),
      ('E0013', 'S0003'),
      ('E0014', 'S0004'),
      ('E0015', 'S0005'),
      ('E0016', 'S0006'),
      ('E0017', 'S0007'),
      ('E0018', 'S0008'),
      ('E0019', 'S0009'),
      ('E0020', 'S0010');
```

d. Insert sample of data to ACCOUNT1

```
INSERT INTO ACCOUNT1
VALUES
```

```
      ('B0001', 'David Wallace', '2015-12-12', 'wallace1@gmail.com',
'67 Euclid Ave, Cleveland,OH'),
      ('B0002', 'Jo Nguyen', '2018-08-04','nguyen1@gmail.com', '424
Overlook, San Francisco,CA'),
      ('B0003', 'Mary Adcock', '2017-06-01', 'adcockt123@gmail.com',
'231 Pope st, Athens, GA'),
      ('B0004', 'Harrison Yi', '2017-08-04',  'yi1@gmail.com', '3422
May Ave, Boston,MA'),
      ('B0005', 'Yoona Im', '2018-01-12', 'im123@gmail.com','78 Cedar
Rd, San Antonio,TX'),
      ('B0006', 'Jack Lim', '2019-04-21', 'lim023@gmail.com', '603
Harley Dr, Columbus, OH'),
      ('B0007', 'David Wang', '2020-10-19', 'wang92@gmail.com', '110
Tibet Rd, Columbus, OH'),
      ('B0008', 'Brian Davis', '2019-03-20', 'davis3@gmail.com', '117 E
Weber Rd, Columbus, OH'),
      ('B0009', 'Dante Simonetti', '2018-05-14', 'simon192@gmail.com',
'128 Crestview Rd, Columbus, OH'),
      ('B0010', 'Robert Mori', '2019-03-20', 'mori999@gmail.com', '72 E
Tulane Rd, Columbus, OH'),
      ('S0001', 'Cade Sbrocco', '2016-07-08', 'sbroccocade@gmail.com',
'5983 Mystic Rdg, Erie, PA'),
      ('S0002', 'Nick Jones', '2018-11-16', 'nwjones@gmail.com', '324
Forbes Ave, Pittsburgh, PA'),
      ('S0003', 'Cory Branton', '2017-05-25', 'cbranton7@gmail.com',
'231 5th Ave, Pittsburgh, PA'),
      ('S0004', 'Luca Lavezzo', '2016-03-12', 'llavez@gmail.com', '423
State St, Cincinnati, OH'),
      ('S0005', 'Brandon Mannley', '2016-11-15', 'bman@gmail.com', '231
Euclid Ave, Cleveland, OH'),
      ('S0006', 'Ozzy Osburne', '2016-11-15', 'ozzy@gmail.com', '254
Euclid Ave, Cleveland, OH'),
      ('S0007','Steve Harris', '2016-11-15', 'ss@gmail.com', '531
Euclid Ave, Cleveland, OH'),
      ('S0008','Sid Vicious', '2016-11-15', 'sd@gmail.com', '666 Euclid
Ave, Cleveland, OH'),
      ('S0009','Johnny Rotten', '2016-11-15', 'jr@gmail.com', '789
Harley Dr, Columbus, OH'),
      ('S0010','Jeff Hannyman', '2016-11-15', 'jh666@gmail.com', '555
Lane Ave, Cleveland, OH');
```

e. Insert sample of data to ACCOUNT2

```
INSERT INTO ACCOUNT2
VALUES
      ('wallace1@gmail.com', 6142847278),
```

```
('nguyen1@gmail.com', 3124675746),
('adcockt123@gmail.com',2162343212),
('yi1@gmail.com', 61473848383),
('im123@gmail.com', 4064738576),
( 'lim023@gmail.com', 6149322812),
('wang92@gmail.com',6143823723),
('davis3@gmail.com', 6143820221),
('simon192@gmail.com', 6140392810),
('mori999@gmail.com', 6149302812),
('sbroccocade@gmail.com', 8149202721),
('nwjones@gmail.com', 8142329090),
('cbranton7@gmail.com', 8143235400),
('llavez@gmail.com', 2341220343),
('bman@gmail.com', 9124506783),
('ozzy@gmail.com', 6147957783),
('ss@gmail.com', 6147958787),
('sd@gmail.com', 6147958866),
('jr@gmail.com', 6147957979),
('jh666@gmail.com', 6147957414);
```

f. Insert sample data to PHOTOS_ACCOUNT

```
INSERT INTO PHOTOS_ACCOUNT
VALUES
      ('B0001', 123),
      ('B0002', 124),
      ('B0003', 125),
      ('B0004', 126),
      ('B0005', 127),
      ('B0006', 128),
      ('B0007', 129),
      ('B0008', 130),
      ('B0009', 131),
      ('B0010', 132),
      ('S0001', 133),
      ('S0002', 134),
      ('S0003', 235),
      ('S0004', 336),
      ('S0005', 437),
      ('S0006', 538),
      ('S0007', 939),
      ('S0008', 340),
      ('S0009', 341),
      ('S0010', 642);
```

g. Insert sample of data to STORE

```
INSERT INTO STORE
VALUES
      ('Go Shop', '2018-01-20', NULL, NULL, NULL, NULL, 6142738273, 4),
      ('Luxx', '2017-05-23', NULL,NULL,NULL,NULL, 6142842842, 5),
      ('Nava', '2019-09-20', NULL,NULL,NULL,NULL, 6141192288, 3),
      ('FaciArt', '2020-05-04', NULL, NULL, NULL,NULL, 6142902008, 4),
      ('Ama Accessory', '2018-12-08', NULL, NULL, NULL,NULL,
6147262008, 5),
      ('Happy Drive', '2016-12-12', NULL, NULL, NULL, NULL, 6148301820,
5),
      ('ABC Mart', '2018-11-03', NULL, NULL, NULL, NULL, 6143812910,
3),
      ('GCG Gaming', '2020-3-25', NULL, NULL, NULL, NULL, 6149301572,
5),
      ('Sweet Spot', '2017-5-29', NULL, NULL, NULL, NULL, 6147292932,
4),
      ('Holy Moly', '2019-10-01', NULL, NULL, NULL, NULL, 6148832933,
4),
      ('Super Duper', '2017-09-18', NULL, NULL, NULL, NULL, 8147789200,
4),
      ('Party Place', '2017-03-08', NULL, NULL, NULL, NULL, 8148882227,
3),
      ('Great Deals', '2019-10-31', NULL, NULL, NULL, NULL, 8143239456,
5),
      ('Stuff 4 U', '2017-12-01', NULL, NULL, NULL, NULL, 8143332290,
3),
      ('Here 2 Help', '2018-02-08', NULL, NULL, NULL, NULL, 8146578902,
2),
      ('Holy Diver', '2017-06-06', NULL, NULL, NULL, NULL, 6147897777,
4),
      ('Iron Maiden', '2019-02-14', NULL, NULL, NULL, NULL, 6146666666,
3),
      ('Highway 2 Hell', '2018-02-14', NULL, NULL, NULL, NULL,
814666666, 5),
      ('Die Die My Darling', '2017-02-14', NULL, NULL, NULL, NULL,
8145554444, 3),
      ('Dead Rose', '2017-02-14', NULL, NULL, NULL, NULL, 8148888888,
2);
```

h. Insert sample of data to PHOTOS_STORE

```
INSERT INTO PHOTOS_STORE
VALUES
      ('Go Shop', 3423),
      ('Luxx', 4324),
```

```
('FaciArt', 7452),
('FaciArt', 47234),
('Ama Accessory', 4832),
('Happy Drive', 19292),
('ABC Mart', 2911),
('GCG Gaming', 5838),
('Sweet Spot', 39973),
('Holy Moly', 22283),
('Super Duper', 30192),
('Party Place', 32412),
('Stuff 4 U',11111),
('Great Deals', 32111),
('Here 2 Help', 93241),
('Holy Diver',00001),
('Iron Maiden',00001),
('Highway 2 Hell',00001),
('Die Die My Darling',00001),
('Dead Rose',00001);
```

i. Insert sample of data to BUYER

```
INSERT INTO BUYER
VALUES
          ('B0001',NULL,'67 Euclid Ave, Cleveland,OH', 100),
          ('B0002',NULL,'424 Overlook, San Francisco,CA', 200),
          ('B0003',NULL,'231 Pope st, Athens, GA', 106),
          ('B0004',NULL,'3422 May Ave, Boston,MA', 55),
          ('B0005',NULL,'78 Cedar Rd, San Antonio,TX', 66),
          ('B0006',NULL, '603 Harley Dr, Columbus, OH', 78),
          ('B0007',NULL, '110 Tibet Rd, Columbus, OH', 79),
          ('B0008',NULL, '117 E Weber Rd, Columbus, OH', 85),
          ('B0009',NULL, '128 Crestview Rd, Columbus, OH', 60),
          ('B0010',NULL, '72 E Tulane Rd, Columbus, OH', 100);
```

j. Insert sample of data to SELLER

```
INSERT INTO SELLER
VALUES
          ('S0001', NULL),
          ('S0002', 4039293828389232),
          ('S0003', 2839392339232913),
          ('S0004', 3923392349133904),
          ('S0005', 8937918339183813),
          ('S0006', 1939411118392220),
          ('S0007', 1939402018392220),
          ('S0009', 1922391004927281),
```

```
        ('S0010', NULL);
```

k. Insert sample of data to OPEN

```
INSERT INTO OPEN
VALUES
      ('Go Shop', 'S0001'),
      ('Luxx', 'S0002'),
      ('Nava', 'S0003'),
      ('FaciArt', 'S0004'),
      ('Ama Accessory', 'S0005'),
      ('Happy Drive', 'S0006'),
      ('ABC Mart', 'S0007'),
      ('GCG Gaming', 'S0008'),
      ('Sweet Spot', 'S0009'),
      ('Holy Moly', 'S00010'),
      ('Super Duper', 'S0010'),
      ('Party Place', 'S0003'),
      ('Stuff 4 U', 'S0002'),
      ('Great Deals', 'S0001'),
      ('Here 2 Help', 'S0009'),
      ('Holy Diver', 'S0009'),
      ('Iron Maiden', 'S0006'),
      ('Highway 2 Hell', 'S0005'),
      ('Die Die My Darling', 'S0004'),
      ('Dead Rose', 'S0001');
```

l. Insert sample of data to PAYMENTS

```
INSERT INTO PAYMENTS
VALUES
      (1, NULL, 4004103013933813,NULL,NULL,NULL,'Credit card', NULL,
NULL, 'B0001'),
      (2, NULL, NULL, 3847283238423843, NULL, NULL, 'Debit card', NULL,
NULL,'B0002'),
      (3, NULL,2374274228322831, NULL, NULL, NULL, 'Credit card', NULL,
NULL, 'B0003'),
      (4, NULL,2374274228322831, NULL, NULL, NULL,'Credit card', NULL,
NULL, 'B0004'),
      (5, NULL,2374274228322831, NULL, NULL, NULL,'Credit card', NULL,
NULL, 'B0005'),
      (6, NULL, NULL, NULL, NULL, 21928371314,'Bank transfer', NULL,
NULL, 'B0006'),
      (7, NULL, NULL, 4117774029313922, NULL, NULL, 'Debit card', NULL,
NULL,'B0007'),
```

```
      (8, NULL, NULL, 1929472222915974, NULL, NULL, 'Debit card', NULL,
NULL,'B0008'),
      (9, 129482716292, NULL, NULL, NULL, NULL, 'Bank check', NULL,
NULL,'B0009'),
      (10, NULL, NULL, NULL, NULL, 12293928720, 'Bank transfer', NULL,
NULL, 'B0010'),
      (11, 129482713292, NULL, NULL, NULL, NULL, 'Bank check', NULL,
NULL, 'B0001'),
      (12, NULL, NULL, NULL, NULL, 12293928720,'Bank transfer', NULL,
NULL,'B0002'),
      (13, NULL, NULL, 4147774069313922, NULL, NULL,'Debit card', NULL,
NULL, 'B0003'),
      (14, NULL, NULL, 1929332222915974, NULL, NULL,'Debit card', NULL,
NULL, 'B0004'),
      (15, NULL, NULL, NULL, NULL, 66666663720,'Bank transfer', NULL,
NULL, 'B0005'),
      (16,NULL, NULL, NULL, NULL, 21928376666,'Bank transfer', NULL,
NULL, 'B0006'),
      (17,NULL, NULL, NULL, NULL, 2293778720,'Bank transfer', NULL,
NULL, 'B0007'),
      (18,NULL, NULL, NULL, NULL, 2293924440,'Bank transfer', NULL,
NULL, 'B0008'),
      (19,NULL, NULL, NULL, NULL, 2277728720,'Bank transfer', NULL,
NULL, 'B0009'),
      (20,NULL, NULL, NULL, NULL, 2266668720,'Bank transfer', NULL,
NULL, 'B0010');
```

m. Insert sample of data to ORDER_

```
INSERT INTO ORDER_
VALUES
      ('T0001', NULL, '2018-10-29','B0001', 1),
      ('T0002', NULL, '2018-12-02', 'B0002', 2),
      ('T0003', NULL, '2019-04-24', 'B0002', 2),
      ('T0004', NULL, '2019-10-09', 'B0001', 11),
      ('T0005', NULL, '2020-10-29', 'B0003', 3),
      ('T0006', NULL, '2020-10-29', 'B0006', 6),
      ('T0007', NULL, '2020-05-23', 'B0003', 13),
      ('T0008', NULL, '2020-10-15', 'B0003', 13),
      ('T0009', NULL, '2019-07-10', 'B0009', 9),
      ('T0010', NULL, '2020-10-01', 'B0010', 20),
      ('T0011', NULL, '2020-11-29', 'B0006',16),
      ('T0012', NULL, '2020-04-23', 'B0007', 7),
      ('T0013', NULL, '2020-11-15', 'B0008', 18),
      ('T0014', NULL, '2019-03-10', 'B0009', 9),
      ('T0015', NULL, '2020-08-01', 'B0010', 10),
```

```
      ('T0016', NULL, '2020-10-19', 'B0004', 14),
      ('T0017', NULL, '2020-05-16', 'B0009', 19),
      ('T0018', NULL, '2020-09-20', 'B0010',10),
      ('T0019', NULL, '2019-03-11', 'B0003', 3),
      ('T0020', NULL, '2020-01-01', 'B0007',17);
```

n. Insert sample of data to HAS

```
INSERT INTO HAS
VALUES
      ('T0001', 'P0003',12),
      ('T0002', 'P0005', 10),
      ('T0003', 'P0001', 31),
      ('T0004', 'P0002', 20),
      ('T0005', 'P0003', 40),
      ('T0006', 'P0006', 55),
      ('T0007', 'P0009', 47),
      ('T0008', 'P0010', 48),
      ('T0009', 'P0007', 87),
      ('T0010', 'P0008', 33),
      ('T0011', 'P0011', 30),
      ('T0012', 'P0019', 2),
      ('T0012', 'P0002', 18),
      ('T0013', 'P0013', 34),
      ('T0013', 'P0001', 1),
      ('T0014', 'P0002', 32),
      ('T0014', 'P0020', 24),
      ('T0015', 'P0005', 32),
      ('T0016', 'P0002', 60),
      ('T0017', 'P0003', 1),
      ('T0018', 'P0009', 20),
      ('T0019', 'P0010', 17),
      ('T0020', 'P0012', 49);
```

o. Insert sample of data to PRODUCT

```
INSERT INTO PRODUCT
VALUES
      ('P0001', 'Go Shop', 'je3',20, NULL, 200, 'txt', 4, NULL,
'S0001'),
      ('P0002', 'Nava', 'rep',4, NULL, 500, 'txt', 2, NULL, 'S0004'),
      ('P0003', 'Luxx', 'plant',10, NULL, 100, 'txt', 4, NULL,
'S0002'),
      ('P0004', 'Go Shop', 'snack',40, NULL, 800, 'txt', 5, NULL,
'S0001'),
      ('P0005', 'Ama Accessory', 'vi',12, NULL, 500, 'txt', 3, NULL,
'S0005'),
```

```
      ('P0006', 'Super Duper', 'new_type', 100, NULL, 500, '.pdf', 5,
NULL, 'S0010'),
      ('P0007', 'Party Place', '3d_rty', 5, NULL, 1000, '.png', 4,
NULL, 'S0003'),
      ('P0008', 'Stuff 4 U', 'Jk1234', 20, NULL, 15, '.mp3', 3, NULL,
'S0002'),
      ('P0009', 'Great Deals', 'gr8_file', 450, NULL, 0, '.txt.', 5,
NULL, 'S0001'),
      ('P0010', 'Here 2 Help', '1001010', 1000, NULL, 1, '.bin',
5,NULL, 'S0009'),
      ('P0011', 'Holy Diver', 'DIO', 666, NULL, 0, '.txt.', 5, NULL,
'S0009'),
      ('P0012', 'Holy Diver', 'Computer as God', 777, NULL, 0, '.txt.',
5, NULL, 'S0009'),
      ('P0013', 'Die Die My Darling', 'Misfits', 1000, NULL, 0,
'.txt.', 4, NULL, 'S0004'),
      ('P0014', 'Iron Maiden', 'Fear of the dark', 789, NULL, 0,
'.bin', 3, NULL, 'S0006'),
      ('P0015', 'Highway 2 Hell', 'AC-DC', 1010, 'Go Dying', 0, '.bin',
2, NULL, 'S0005'),
      ('P0016', 'Happy Drive', 'handle_93', 100, NULL, 15, '.txt', 4,
NULL, 'S0007'),
      ('P0017', 'ABC Mart', 'abcsh01', 50, NULL, 500, '.txt', 4, NULL,
'S0002'),
      ('P0018', 'GCG Gaming', 'sc2_012',60, NULL, 100, '.txt',4, NULL,
'S0003'),
      ('P0019', 'Sweet Spot', 't_1032', 10, NULL, 10, '.txt',2, NULL,
'S0007'),
      ('P0020', 'Holy Moly', 'hm_39', 400, NULL, 0, '.txt', 3, NULL,
'S0010');
```

p. Insert sample of data to PHOTOS_PRODUCT

```
INSERT INTO PHOTOS_PRODUCT
VALUES
      ('P0001', 02012),
      ('P0002', 00002),
      ('P0003', 00293),
      ('P0004', 01290),
      ('P0005', 03710),
      ('P0006', 00456),
      ('P0007', 00457),
      ('P0008', 00458),
      ('P0009', 00459),
      ('P0010', 00470),
```

```
      ('P0011', 00001),
      ('P0012', 00002),
      ('P0013', 00012),
      ('P0014', 00022),
      ('P0015', 01300),
      ('P0016', 03812),
      ('P0017', 03813),
      ('P0018', 03814),
      ('P0019', 03815),
      ('P0020', 03816);
```

q. Insert sample of data to KEYWORD

```
INSERT INTO KEYWORDS
VALUES
      ('P0001', 'ahd'),
      ('P0002', 'ha'),
      ('P0003', 'den'),
      ('P0004', 'sweet'),
      ('P0005','hat');
      ('P0006', 'X'),
      ('P0007', 'helpful'),
      ('P0008', 'useful'),
      ('P0009', 'needed'),
      ('P0010', 'new'),
      ('P0011', 'dio'),
      ('P0012', 'black sabbath'),
      ('P0013', 'punk'),
      ('P0014', 'piece of mind'),
      ('P0015','metal'),
      ('P0016', 'alloy'),
      ('P0017', 'leather'),
      ('P0018', 'game'),
      ('P0019', 'sweet'),
      ('P0020', 'holy');
```

r. Insert sample of data to FEEDBACK

```
INSERT INTO FEEDBACK
VALUES
      ('F0001', NULL, 4,'T0001'),
      ('F0002', NULL,  4,'T0002'),
      ('F0003', NULL, 5, 'T0003'),
      ('F0004', NULL, 2, 'T0004'),
```

```
('F0005', NULL, 4,'T0005'),
('F0006', 'Nice Item', 4,'T0006'),
('F0007', 'Worth the money', 4,'T0007'),
('F0008', 'Great seller', 5,'T0008'),
('F0009', NULL, 2, 'T0009'),
('F0010', NULL,4, 'T0010'),
('F0011', 'I do not like it', 2, 'T0011'),
('F0012', 'This is so cool', 5,'T0012'),
('F0013', 'Customer service is really bad', 1, 'T0013'),
('F0014', 'Not bad at all', 3, 'T0014'),
('F0015', 'Seller is so kind', 5, 'T0015'),
('F0016','Not bad at all', 4, 'T0016'),
('F0017', 'Not bad at all', 4, 'T0017'),
('F0018', 'Not bad at all',4, 'T0018'),
('F0019', 'Not bad at all', 3, 'T0019'),
('F0020', 'Very good', 5, 'T0020');
```

2.1.3 SQL QUERIES

Queries From part 1/ section 2

a. Find the buyer who has purchased the most IP Items and the total number of IP Items they have purchased

```
SELECT A.Name,    COUNT(*)
FROM ACCOUNT1 AS A, ORDER_ AS ORD, HAS AS HA, PRODUCT AS PR
WHERE A.Account_id = ORD.Account_id AND ORD.Transaction_id =
HA.Transaction_id AND HA.Product_id = PR.Product_id
GROUP BY A.Name
HAVING COUNT(*)=
(SELECT MAX(BC)
FROM (SELECT   COUNT(*) AS BC
            FROM ACCOUNT1 AS A, ORDER_ AS ORD, HAS AS HA, PRODUCT AS PR
      WHERE A.Account_id = ORD.Account_id AND ORD.Transaction_id =
HA.Transaction_id AND HA.Product_id = PR.Product_id
            GROUP BY A.Name) );
```

b. Give all the buyers who purchased a IP Item by a given seller and the names of the IP Items they purchased, seller given by Account_id = 'S0003'

```
SELECT b.*, p.Name
FROM PRODUCT AS p, BUYER AS b, ORDER_ AS o, HAS AS h, ACCOUNT1 AS a
```

```
WHERE p.Account_id = 'S0003' AND  h.Transaction_id = o.Transaction_id AND
o.Account_id = 'B0001' AND h.Product_id = p.Product_id AND b.Account_id =
a.Account_id;
```

Queries From Checkpoints (Problem 3)

a. Find the titles of all IP Items by a given Seller that cost less than $10 (you choose how to designate the seller) *Seller designated by Account_id = S_id*

Relational Algebra:

RESULT ← $\pi_{Title}(\sigma_{Price<10}(PRODUCT)*\sigma_{Account\_id=S\_id}(SELLER))$

SQL:

```
SELECT      p.Name
FROM        PRODUCT AS p
WHERE       p.Account_id = 'S0001' AND p.Price < 10;
```

b. Give all the titles and their dates of purchase made by given buyer (you choose how to designate the buyer) *Buyer designated by Account_id = B_id*

Relational Algebra:

BUYER_ORDER ←

$(\sigma_{Account\_id=B\_id}(BUYER)\bowtie_{Account\_id=Account\_id}(ORDER\_))\bowtie_{Transaction\_id=Transaction\_id}HAS)$

RESULT ← $\pi_{Title,Date}(BUYER\_ORDER\bowtie_{Product\_id=Product\_id}PRODUCT)$

SQL:

```
SELECT      p.Name, o.Date_of_order
FROM        PRODUCT AS p, ORDER_ AS o, HAS AS h
WHERE       h.Transaction_id = o.Transaction_id AND o.Account_id =
            'B0001' AND h.Product_id = p.Product_id;
```

c. Find the seller names for all sellers with less than 5 IP Items for sale

Relational Algebra:

RESULT ← $\pi_{Name}(ACCOUNT1*(\sigma_{Count\_product\_id<5}(F_{COUNT}$

$_{Product\_id}(SELLER\bowtie_{Account\_id=Account\_id}(PRODUCT)))))$

SQL:

```
SELECT      a.Name
```

```
FROM        PRODUCT AS p, ACCOUNT1 AS a, SELLER AS s
WHERE       s.Account_id = a.Account_id AND p.Account_id = a.Account_id
GROUP BY    s.Account_id
HAVING      count(p.Product_id) < 5;
```

d. Give all the buyers who purchased a IP Item by a given seller and the names of the IP Items they purchased Seller designated by Account_id = S_id

Relational Algebra:

BUYER_ORDER ←

$(\sigma_{Account\_id=B\_id}(BUYER) \bowtie_{Account\_id=Account\_id}(ORDER\_)) \bowtie_{Transaction\_id=Transaction\_id} HAS)$

SELLER_PRODUCT ← PRODUCT*$\sigma_{Account\_id=S\_id}$(SELLER)

RESULT ← $\pi_{Account\_id,title}$(BUYER_ORDER$\bowtie_{Product_id=Product_id}$SELLER_PRODUCT)

SQL:

```
SELECT      a.Name,P.Name
FROM        PRODUCT AS p, ORDER_ AS o, HAS AS h, ACCOUNT1 AS a
WHERE       p.Account_id = 'S0003' AND  h.Transaction_id =
            o.Transaction_id AND o.Account_id = a.Account_id AND
            h.Product_id = p.Product_id;
```

e. Find the total number of IP Items purchased by a single buyer (you choose how to designate the buyer) *Buyer designated by Account_id = B_id*

Relational Algebra:

BUYER_ORDER ←

$(\sigma_{Account\_id=B\_id}(BUYER) \bowtie_{Account\_id=Account\_id}(ORDER\_)) \bowtie_{Transaction\_id=Transaction\_id} HAS)$

RESULT ← $\pi_{Count\_pid}(F_{COUNT\ Product\_id}$(BUYER_ORDER$\bowtie_{Product_id=Product_id}$PRODUCT)))

SQL:

```
SELECT      COUNT(p.Product_id)
FROM        PRODUCT AS p, ORDER_ AS o, HAS AS h
WHERE       h.Transaction_id = o.Transaction_id AND o.Account_id =
            'B0003' AND h.Product_id = p.Product_id;
```

f. Find the buyer who has purchased the most IP Items and the total number of IP Items they have purchased

Relational Algebra:

RESULT ←$\pi_{Account\_id,\ Max\_count\_pid}(F_{MAX\ Count\_pid}(_{Account\_Id}F_{COUNT\ Product\_id}(ORDER\_ \bowtie_{Transaction\_id=Transaction\_id}(HAS))))$

SQL:

```
SELECT     A.Name, COUNT(*)
FROM       ACCOUNT1 AS A, ORDER_ AS ORD, HAS AS HA, PRODUCT AS PR
WHERE      A.Account_id = ORD.Account_id AND ORD.Transaction_id =
           HA.Transaction_id AND HA.Product_id = PR.Product_id
GROUP BY   A.Name
HAVING     COUNT(*)=
                (SELECT MAX(BC)
                FROM (SELECT   COUNT(*) AS BC
                     FROM       ACCOUNT1 AS A, ORDER_ AS ORD, HAS AS
                                HA, PRODUCT AS PR
                     WHERE      A.Account_id = ORD.Account_id AND
                                ORD.Transaction_id = HA.Transaction_id
                                AND HA.Product_id = PR.Product_id
                     GROUP BY   A.Name));
```

Queries From Checkpoints (Problem 4)

a. List all the Sellers (Seller_id) and the number of stores for each seller*/

Relational Algebra:

Result ← πAccount_id, Count_SN((Account_id)F COUNT Store_name

(STORE*(SELLER*OPEN)))

SQL:

```
SELECT     Account_id, COUNT(*)
FROM       OPEN
GROUP BY   Account_id;
```

b. Count the number of unique products that are sold in a store called “Super Duper”*/

Relational Algebra:

Result← πCount_PI(F COUNT Product_id (σStore_name=”Super_Duper”(PRODUCT)))

SQL:

```
SELECT     COUNT(p.Product_id)
FROM       PRODUCT AS p
```

```
WHERE      p.Store_name = 'Super Duper'
```

c. Count the number of BB_EMPLOYEE whose name is "John" */

Relational Algebra:

Name_John←σName=John(BB_EMPLOYEE)

Result ← F COUNT Employee_id (Name_John)

SQL:

```
SELECT     COUNT(b.Employee_id)
FROM       BB_EMPLOYEE1 as b
WHERE      b.Name LIKE 'John%';
```

Queries From Checkpoints (Problem 5)

a. Provide a list of buyer names, along with the total dollar amount each buyer has spent. */

```
SELECT     A.Name, SUM (P.Price*H.Qty)
FROM       ACCOUNT1 AS A, BUYER AS B, ORDER_ AS O, HAS AS H, PRODUCT
AS P
WHERE      A.Account_id=B.Account_id AND B.Account_id=O.Account_id
           AND O.Transaction_id = H.Transaction_id AND
           P.Product_id=H.Product_id
GROUP BY   A.Name;
```

b. Provide a list of buyer names and e-mail addresses for buyers who have spent more than the average buyer.

```
SELECT     A.Name,A.Email
FROM       ACCOUNT1 AS A, BUYER AS B, ORDER_ AS O, HAS AS H, PRODUCT
AS P
WHERE      A.Account_id=B.Account_id AND B.Account_id=O.Account_id AND
O.Transaction_id = H.Transaction_id AND     P.Product_id=H.Product_id
GROUP BY   A.account_id
HAVING           SUM(H.Qty*P.Price)
           >(SELECT   SUM(H.Qty*P.Price)/COUNT(DISTINCT A.Account_id)
             FROM     ACCOUNT1 AS A, BUYER AS B, ORDER_ AS O, HAS AS H,
PRODUCT AS P
             WHERE    A.Account_id=B.Account_id AND
           B.Account_id=O.Account_id AND O.Transaction_id =
           H.Transaction_id AND  P.Product_id=H.Product_id);
```

c. Provide a list of the IP Item names and associated total copies sold to all buyers,

sorted from the IP Item that has sold the most individual copies to the IP Item that has sold the least.

```
SELECT     PR.Name, SUM(HA.QTY)
FROM       ACCOUNT1 AS A, ORDER_ AS ORD, HAS AS HA, PRODUCT AS PR
WHERE      A.Account_id = ORD.Account_id AND ORD.Transaction_id =
HA.Transaction_id AND HA.Product_id = PR.Product_id
GROUP BY   PR.Product_id
ORDER BY   SUM(HA.QTY) DESC
```

d. Provide a list of the IP Item names and associated dollar totals for copies sold to all buyers,

sorted from the IP Item that has sold the highest dollar amount to the IP Item that has sold the smallest.

```
SELECT      P.name,SUM(qty)*P.price AS total
FROM        PRODUCT AS P, HAS AS H
WHERE       P.Product_id=H.Product_id
GROUP BY    P.product_id
ORDER BY    SUM(H.qty)*P.price DESC;
```

e. Find the most popular Seller (i.e. the one who has sold the most IP Items)*/

```
SELECT A2.Name
FROM ACCOUNT1 AS A1, ORDER_ AS ORD1, HAS AS HA1, PRODUCT AS PR1,
ACCOUNT1 AS A2
WHERE A1.Account_id = ORD1.Account_id AND ORD1.Transaction_id =
HA1.Transaction_id AND HA1.Product_id = PR1.Product_id AND
PR1.Account_id = A2.Account_id
GROUP BY PR1.Account_id
HAVING SUM(HA1.QTY)=
        (SELECT MAX(QT)
        FROM (SELECT SUM(HA.QTY) AS QT
              FROM ACCOUNT1 AS A, ORDER_ AS ORD, HAS AS HA, PRODUCT AS
PR
              WHERE A.Account_id = ORD.Account_id AND
ORD.Transaction_id = HA.Transaction_id AND HA.Product_id =
PR.Product_id
              GROUP BY PR.Account_id));
```

f. Find the most profitable seller (i.e. the one who has brought in the most money) */

```
SELECT     A2.Name
FROM       ACCOUNT1 AS A1, ORDER_ AS ORD1, HAS AS HA1, PRODUCT AS PR1,
ACCOUNT1 AS A2
```

```
WHERE      A1.Account_id = ORD1.Account_id AND ORD1.Transaction_id =
HA1.Transaction_id AND HA1.Product_id = PR1.Product_id AND
PR1.Account_id = A2.Account_id
GROUP BY   PR1.Account_id
HAVING     SUM(HA1.QTY*PR1.Price)=
           (SELECT    MAX(QT)
            FROM (SELECT    SUM(HA.QTY*PR.Price) AS QT
                  FROM     ACCOUNT1 AS A, ORDER_ AS ORD, HAS AS HA,
PRODUCT AS PR
                  WHERE A.Account_id = ORD.Account_id AND
ORD.Transaction_id = HA.Transaction_id AND HA.Product_id =
PR.Product_id
                  GROUP BY PR.Account_id));
```

g. Provide a list of buyer names for buyers who purchased anything listed by the most profitable Seller.

```
SELECT     a.name
FROM       ACCOUNT1 AS a, BUYER AS b, HAS AS h, ORDER_ AS O,
           (SELECT    PR1.Account_id, PR1.Product_id
           FROM       ACCOUNT1 AS A1, ORDER_ AS ORD1, HAS AS HA1,
           PRODUCT AS PR1
           WHERE A1.Account_id = ORD1.Account_id AND
           ORD1.Transaction_id = HA1.Transaction_id AND HA1.Product_id
           = PR1.Product_id
           GROUP BY PR1.Account_id
           HAVING SUM(HA1.QTY*PR1.Price)=
           (SELECT MAX(QT)
           FROM (SELECT SUM(HA.QTY*PR.Price) AS QT
           FROM ACCOUNT1 AS A, ORDER_ AS ORD, HAS AS HA, PRODUCT AS PR
           WHERE A.Account_id = ORD.Account_id AND ORD.Transaction_id
           = HA.Transaction_id AND HA.Product_id = PR.Product_id
           GROUP BY PR.Account_id)) ) AS p

WHERE p.Product_id = h.Product_id AND o.Transaction_id =
h.Transaction_id AND o.Account_id = b.Account_id AND b.Account_id =
a.Account_id;
```

h. Provide the list of sellers who listed the IP Items purchased by the buyers who have spent more than the average buyer.*/

```
SELECT A1.Name
FROM ACCOUNT1 as A1, PRODUCT AS p, SELLER AS s, HAS AS h, ORDER_ AS O,
(SELECT         B.*
```

```
 FROM      ACCOUNT1 AS A, BUYER AS B, ORDER_ AS O, HAS AS H, PRODUCT
AS P
 WHERE           A.Account_id=B.Account_id AND
B.Account_id=O.Account_id AND O.Transaction_id = H.Transaction_id AND
      P.Product_id=H.Product_id
 GROUP BY        A.account_id
 HAVING          SUM(H.Qty*P.Price)
                 >(SELECT   SUM(H.Qty*P.Price)/COUNT(DISTINCT
           A.Account_id)
                 FROM  ACCOUNT1 AS A, BUYER AS B, ORDER_ AS O, HAS AS H,
PRODUCT AS P
                 WHERE A.Account_id=B.Account_id AND
           B.Account_id=O.Account_id AND O.Transaction_id =
           H.Transaction_id AND  P.Product_id=H.Product_id)) as b

WHERE p.Product_id = h.Product_id AND o.Transaction_id =
h.Transaction_id AND o.Account_id = b.Account_id AND p.Account_id =
s.Account_id AND A1.Account_id = s.Account_id;
```

2.1.4 SQL INSERT/DELETE from Part 1

A. INSERT Samples

a. Insert items into product

```
INSERT INTO PRODUCT
VALUES
     ('P0001', 'Go Shop', 'je3',20, NULL, 200, 'txt', 4, NULL,
'S0001'),
     ('P0002', 'Nava', 'rep',4, NULL, 500, 'txt', 2, NULL, 'S0004'),
     ('P0003', 'Luxx', 'plant',10, NULL, 100, 'txt', 4, NULL,
'S0002'),
     ('P0004', 'Go Shop', 'snack',40, NULL, 800, 'txt', 5, NULL,
'S0001'),
     ('P0005', 'Ama Accessory', 'vi',12, NULL, 500, 'txt', 3, NULL,
'S0005');
```

b. Insert new stores into store

```
INSERT INTO STORE
VALUES
('Go Shop', '2018-01-20', NULL, NULL, NULL, NULL, 6142738273, 4),
('Luxx', '2017-05-23', NULL,NULL,NULL,NULL, 6142842842, 5),
('Nava', '2019-09-20', NULL,NULL,NULL,NULL, 6141192288, 3),
('FaciArt', '2020-05-04', NULL, NULL, NULL,NULL, 6142902008, 4),
('Ama Accessory, '2018-12-08', NULL, NULL, NULL,NULL, 6147262008,
5),
('Happy Drive', '2016-12-12', NULL, NULL, NULL, NULL, 6148301820,
5);
```

B. DELETE Samples

a. Delete a store name called Go Shop

```
DELETE FROM STORE
WHERE Store_name = "Go Shop"
```

b. Delete a product "P0001" from PRODUCT

```
DELETE FROM PRODUCT
WHERE Product_id = 'P0001'
```

3. Appendix

3.1 CheckPoints

3.1.1 Checkpoint 1

1. List names of all your team members. Provide a paragraph explaining how you have been working as a team under remote setup so far, how you plan to communicate with each other, share work, etc. Any issues related to time differences, technology constraints, etc?

Team Working Description: Since all the team members are living in the same time zone(US eastern), each team member could communicate via GroupMe, Zoom, and Google docs concurrently, without any issues. In project execution, team members will set up a regular virtual online meeting each week to track and discuss project progress. If it is necessary, the team would arrange an in-person meeting with social distance practicing.

2. Based on the requirements given in the project overview, list the entities to be modeled in this database. For each entity, provide a list of associated attributes. Make sure that your design allows for proper handling of buyer /seller interactions such as orders, payments, feedback, and karma points.

Entities	Attributes (and other info)
ACCOUNT	Subclasses: BUYER, SELLER. Attributes: **account_id**, name, **e-mail**, address, phone_number, photo (multivalued), join_date
BUYER	Attributes: default_payment_information, default_delivery_information
SELLER	Attributes: karma_points, bank_information
ORDER	Attributes: **Transaction_ID**, date, payment_information (multivalued), delivery_information
PRODUCT	Attributes: Price, name, availability, keyword (multivalued), **product_ID,** title, description, image (multivalued), file_type, type of payment accepted, rating

FEEDBACK	Subclasses: PRODUCT_FEEDBACK, SELLER_FEEDBACK Attributes: rating, description, Type (Product or Seller), **feedback_id** (defined fully with order number)
PRODUCT_FEEDBACK	Attribute: number bought
SELLER_FEEDBACK	Attribute: orders processed, products sold

3. Based on the requirements given in the project overview, what are the various relationships between entities? (For example, “CUSTOMER entities purchase IP Item entities”).

- Each BUYER can create multiple ORDERs.
- Each ORDER must be created by only one BUYER.
- ORDERs may leave FEEDBACK.
- FEEDBACK is left by one and only one ORDER.
- SELLER_FEEDBACK is left on one and only one ORDER
- SELLER_FEEDBACK is associated with one and only SELLER
- SELLERs can have multiple SELLER_FEEDBACK
- PRODUCT_FEEDBACK is left up to once on each PRODUCT in an ORDER - PRODUCT_FEEDBACK is associated with one and only one PRODUCT - A PRODUCT can have multiple PRODUCT_FEEDBACK
- An ORDER can have up to one FEEDBACK left.
- A SELLER can put up for sale multiple PRODUCTs.
- A PRODUCT can only be put by one and only one SELLER.
- A PRODUCT can be a part of multiple ORDERs.
- ORDER can have multiple PRODUCTs.

4. Propose at least two additional entities that it would be useful for this database to model beyond the scope of the project requirements. Provide a list of possible attributes for the additional entities and possible relationships they may have with each other and the rest of the entities in the database. Give a brief, one sentence rationale for why adding these entities would be interesting/useful to the stakeholders for this database project.

Additional entities:

Store: Sellers would set up many stores(personal owned or multi-owner) based on the product they want to sell.

Entities	Attributes

STORE	Name, description, banner, bio, photos, url, phone number, creation date
BB_EMPLOYEE	Employee_ID, e-mail, phone number, name, salary, schedule, join_date

Additional relationships:

- A STORE can have many PRODUCTS
- A PRODUCT can be in only one STORES
- A SELLER can open or join many STORES
- A STORE must be joined by more than one SELLERS
- A BUYER may contact a BB_EMPLOYEE
- A BB_EMPLOYEE may be contacted by multiple BUYERs and SELLERS - A BUYER may contact multiple BB_EMPLOYEEs

5. Give at least four examples of some informal queries/reports that it might be useful for this database might be used to generate. Include one example for each of the additional entities you proposed in question 3 above.

- Store Selling Report: (Product Name, Product ID, Number of Orders per product, Time Frame(start/end), Total Sell($))
- Buying History: (Transaction ID, Product Name, Product Price, Number of Orders per product,Time Frame(start/end), Total Spending($))
- Product/Store Feedback List: Buyer Name, Date, rating, comment, Time Frame(start/end), average rating, Overall rating)
- Buyer Guides (related products): (Store Name, Store Banner, Store Description, URL, overall rating)
- Employee Information: e-mail, name, phone number, schedule
- Store Information: List of products sold, list of sellers in store, overall rating of sellers, overall rating of products

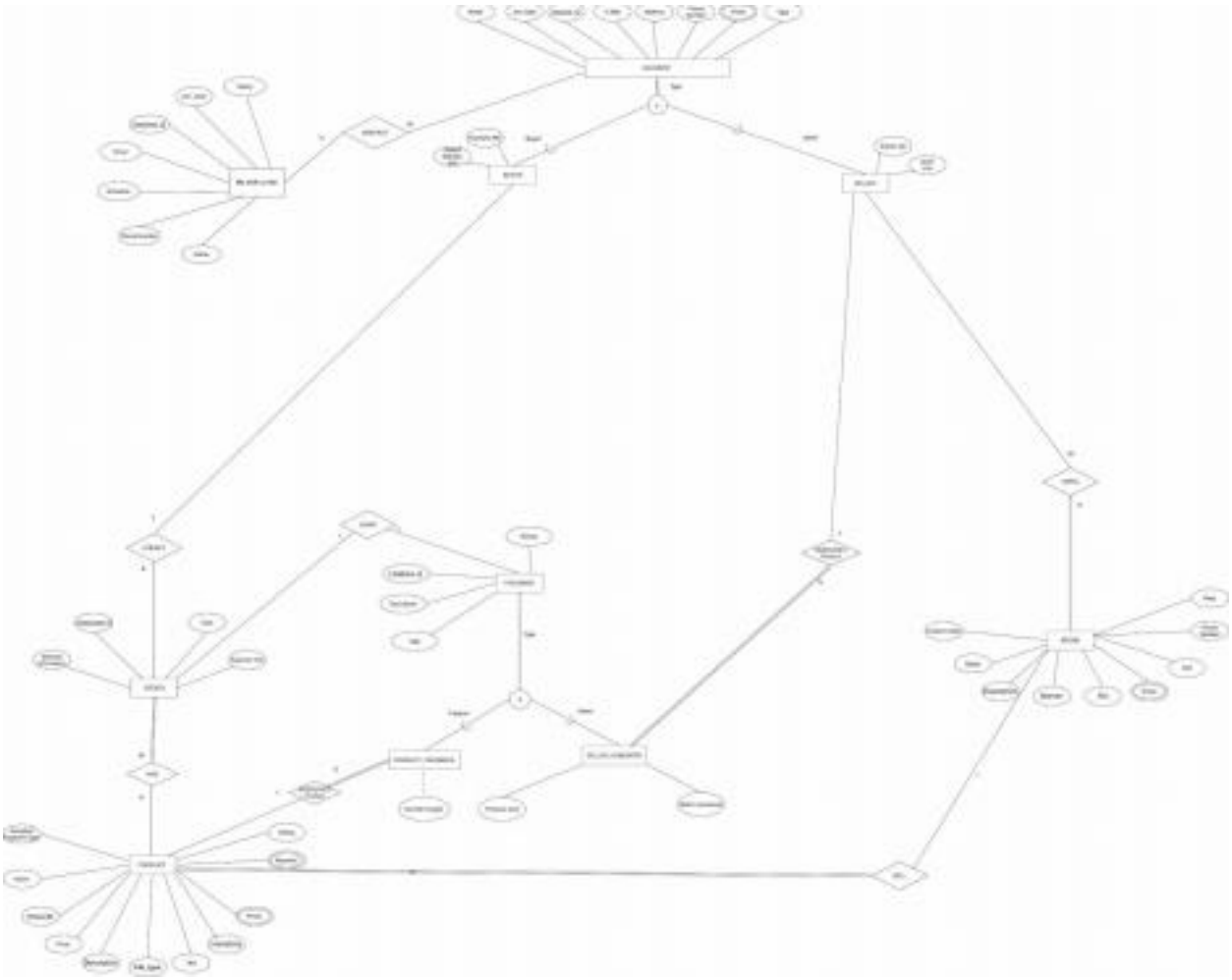
6. Suppose we want to add a new IP Item to the database. How would we do that given the entities and relationships you've outlined above? Is it possible to add up to five images for the IP Item? Is it possible for the IP Item to be purchased by more than one Payment Type? Is it possible for the Buyer to purchase IP Items from multiple Sellers at one time? Can a Buyer leave feedback on multiple items in the Seller's store? Explain how your model supports these possibilities. If it does not, make changes that allow your design to support all these requirements.

- A new IP item would be a PRODUCT, so a PRODUCT instance would be created with a unique product ID.
- Yes, a PRODUCT has a multivalued attribute for photos.
- Yes, an ORDER has a multivalued payment information attribute
- Yes, an ORDER can consist of multiple PRODUCTs.
- Yes, if a BUYER places an ORDER, they are able to leave FEEDBACK on any PRODUCT in the ORDER.

7. Determine at least three other informal update operations and describe what entities would need to have attributes altered and how they would need to be changed given your above descriptions. Include one example for each of the additional entities you proposed in question 4 above.

- A STORE wants to change their landing page. Then their url attribute would be updated. - A BB_EMPLOYEE gets a raise. Then their salary attribute would be updated. - A SELLER receives feedback on an order filled. Then their karma points attribute would be updated.

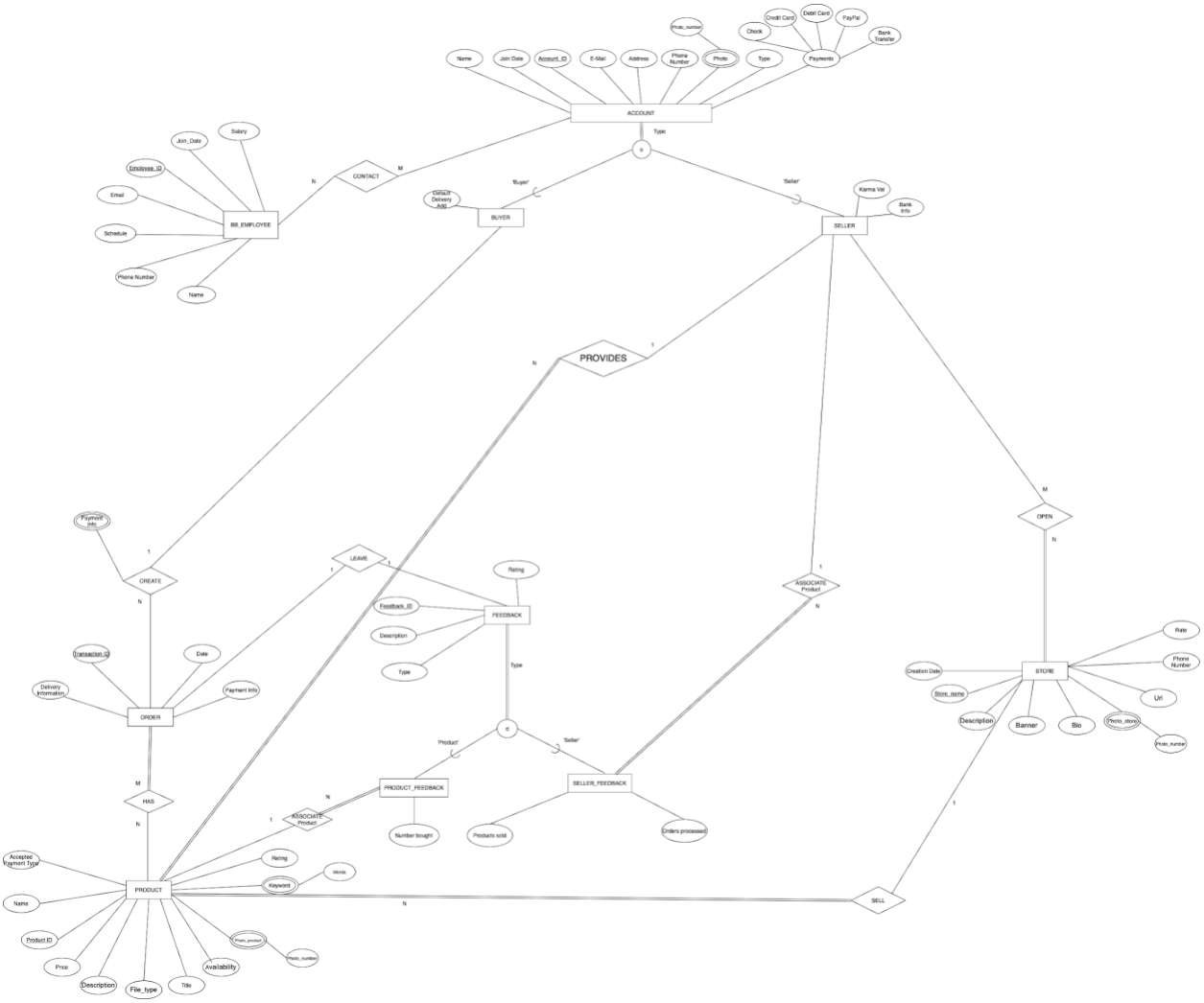
8. Provide an ER diagram for your database. Make sure you include all of the entities and relationships you determined in the questions above INCLUDING the entities for question 4 above, and remember that EVERY entity in your model needs to connect to another entity in the model via some kind of relationship. You can use draw.io for your diagram. If drawing on paper, make sure that your drawing is clear and neat. Ensure that you use a proper notation and include a legend.



3.1.2 Checkpoint 2

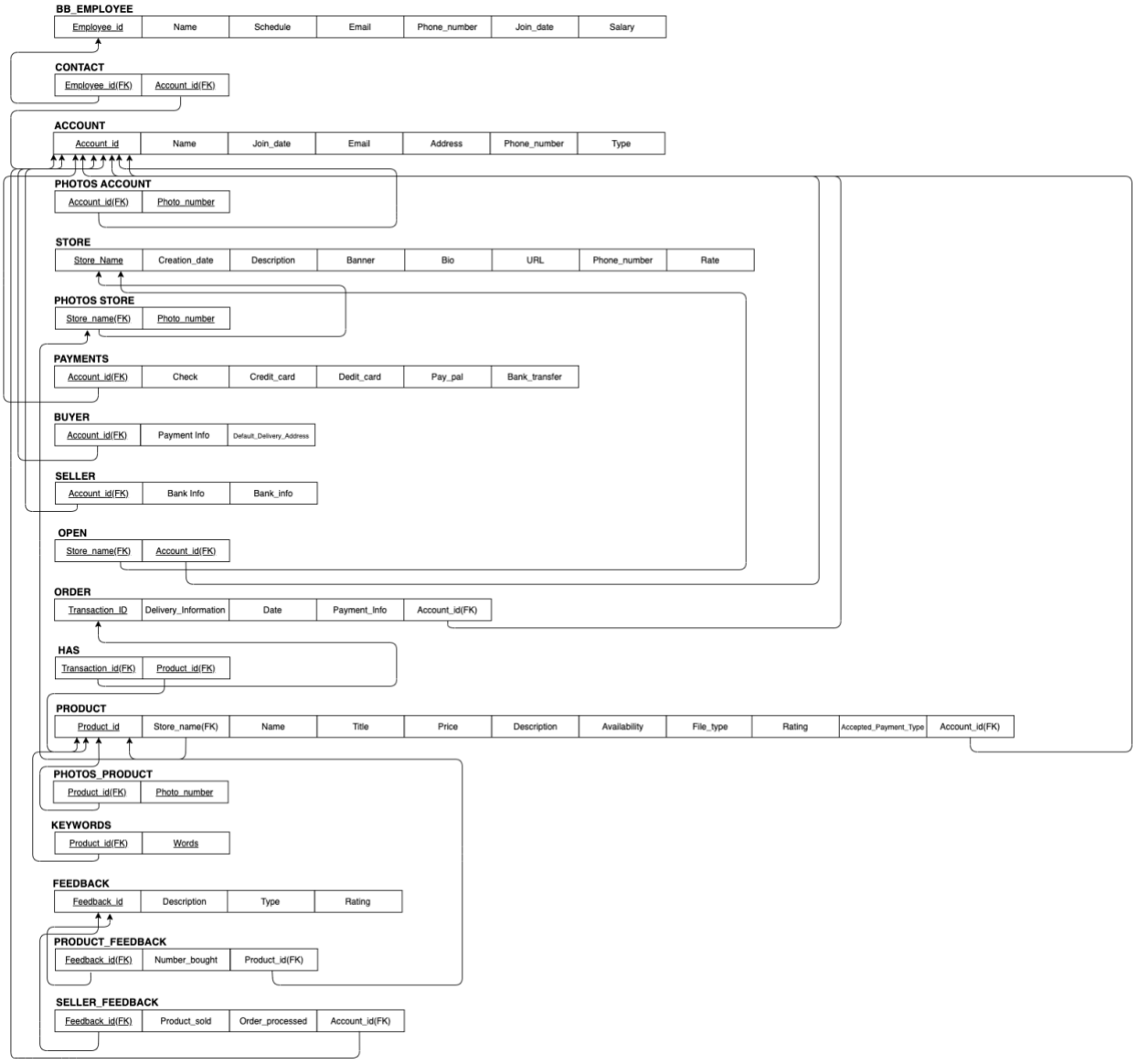
1.

Provide our ER Model (PNG file)



2.

Provide our Relationship Table (PNG file)



3 Given your relational schema, provide the relational algebra to perform the following queries. If your schema cannot provide answers to these queries, revise your ER Model and your relational schema to contain the appropriate information for these queries:

a. Find the titles of all IP Items by a given Seller that cost less than $10 (you choose how to designate the seller)
Seller designated by Account_id = S_id

$RESULT \leftarrow \pi_{Title}(\sigma_{Price<10}(PRODUCT) * \sigma_{Account\_id=S\_id}(SELLER))$

b. Give all the titles and their dates of purchase made by given buyer (you choose how to designate the buyer)
Buyer designated by Account_id = B_id

$BUYER\_ORDER \leftarrow (\sigma_{Account\_id=B\_id}(BUYER) \bowtie_{Account\_id=Account\_id}(ORDER)) \bowtie_{Transaction\_id=Transaction\_id} HAS)$

$RESULT \leftarrow \pi_{Title,Date}(BUYER\_ORDER \bowtie_{Product\_id=Product\_id} PRODUCT)$

c. Find the seller names for all sellers with less than 5 IP Items for sale

$RESULT \leftarrow \pi_{Name}(ACCOUNT * (\sigma_{Count\_product\_id<5}(F_{COUNT\ Product\_id}(SELLER \bowtie_{Account\_id=Account\_id}(PRODUCT)))))$

d. Give all the buyers who purchased a IP Item by a given seller and the names of the IP Items they purchased
Seller designated by Account_id = S_id

$BUYER\_ORDER \leftarrow (\sigma_{Account\_id=B\_id}(BUYER) \bowtie_{Account\_id=Account\_id}(ORDER)) \bowtie_{Transaction\_id=Transaction\_id} HAS)$

$SELLER\_PRODUCT \leftarrow PRODUCT * \sigma_{Account\_id=S\_id}(SELLER)$

$RESULT \leftarrow \pi_{Account\_id,title}(BUYER\_ORDER \bowtie_{Product\_id=Product\_id} SELLER\_PRODUCT)$

e. Find the total number of IP Items purchased by a single buyer (you choose how to designate the buyer)
Buyer designated by Account_id = B_id

$BUYER\_ORDER \leftarrow (\sigma_{Account\_id=B\_id}(BUYER) \bowtie_{Account\_id=Account\_id}(ORDER)) \bowtie_{Transaction\_id=Transaction\_id} HAS)$

$RESULT \leftarrow \pi_{Count\_pid}(F_{COUNT\ Product\_id}(BUYER\_ORDER \bowtie_{Product\_id=Product\_id} PRODUCT)))$

f. Find the buyer who has purchased the most IP Items and the total number of IP Items they have purchased

$RESULT \leftarrow \pi_{Account\_id,\ Max\_count\_pid}(F_{MAX\ Count\_pid}(_{Account\_Id}F_{COUNT\ Product\_id}(ORDER \bowtie_{Transaction\_id=Transaction\_id}(HAS))))$

4.
Three additional interesting queries in plain English and also relational algebra. Your queries should include at least one of these:
a. outer joins
b. aggregate function
c. “extra” entities from CP01

- List all the Sellers (Seller_id) and the number of stores for each seller

Result ← $_{(Account\_id)}F_{COUNT\ Store\_name}$ (STORE*(SELLER*OPEN))

- Count the number of products that are sold in a store called “CSE3241”

Cse_3241← $\sigma_{Store\_name=CSE3241}$(STORE)

Cse_3241_products←PRODUCT*Cse_3241

Result ← $F_{COUNT\ Product\_id}$ (Cse_3241_products)

- Count the number of BB_EMPLOYEE whose name is “John”

Name_John←$\sigma_{Name=John}$(BB_EMPLOYEE)

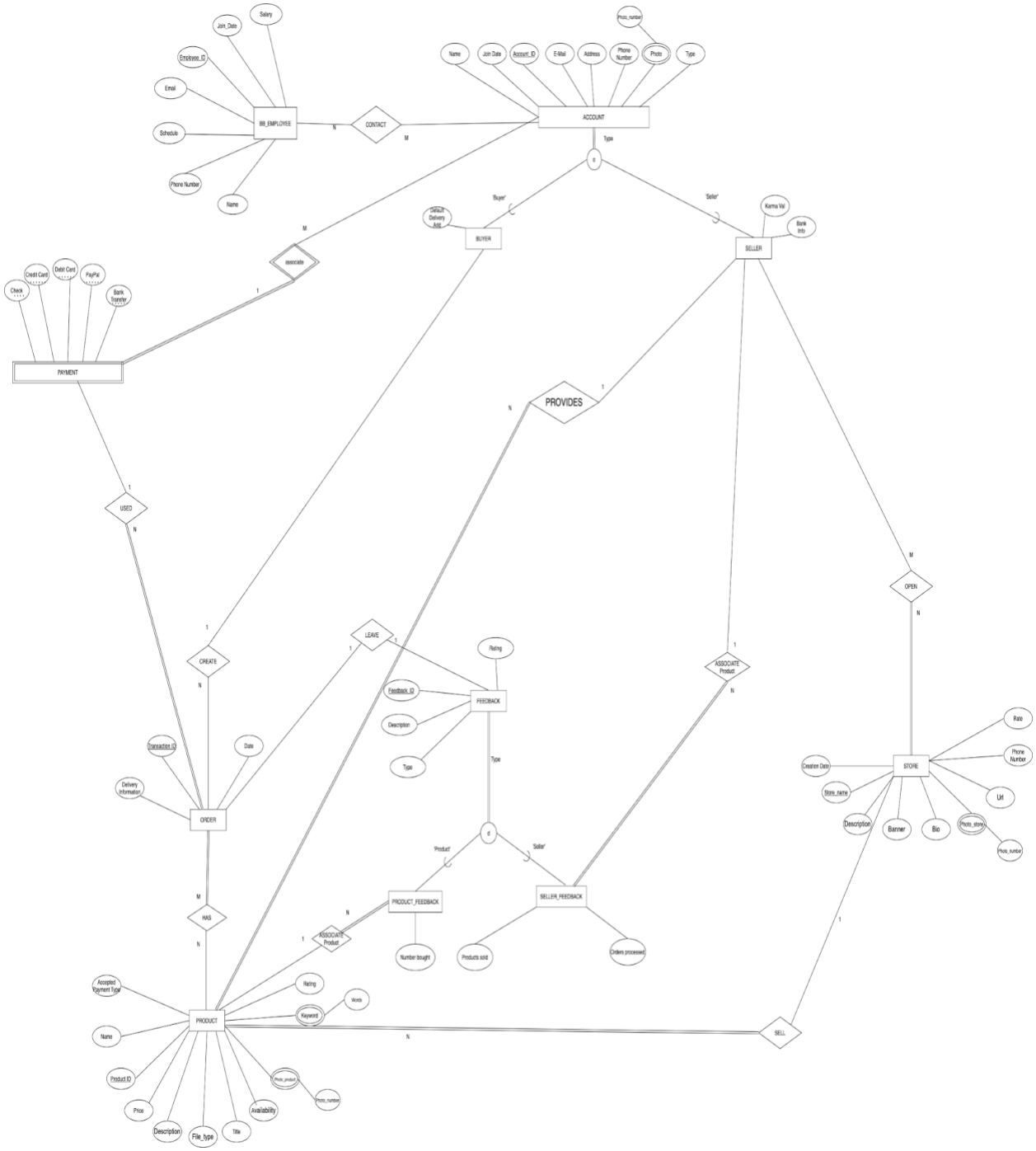
Result ← $F_{COUNT\ Employee\_id}$ (Name_John)

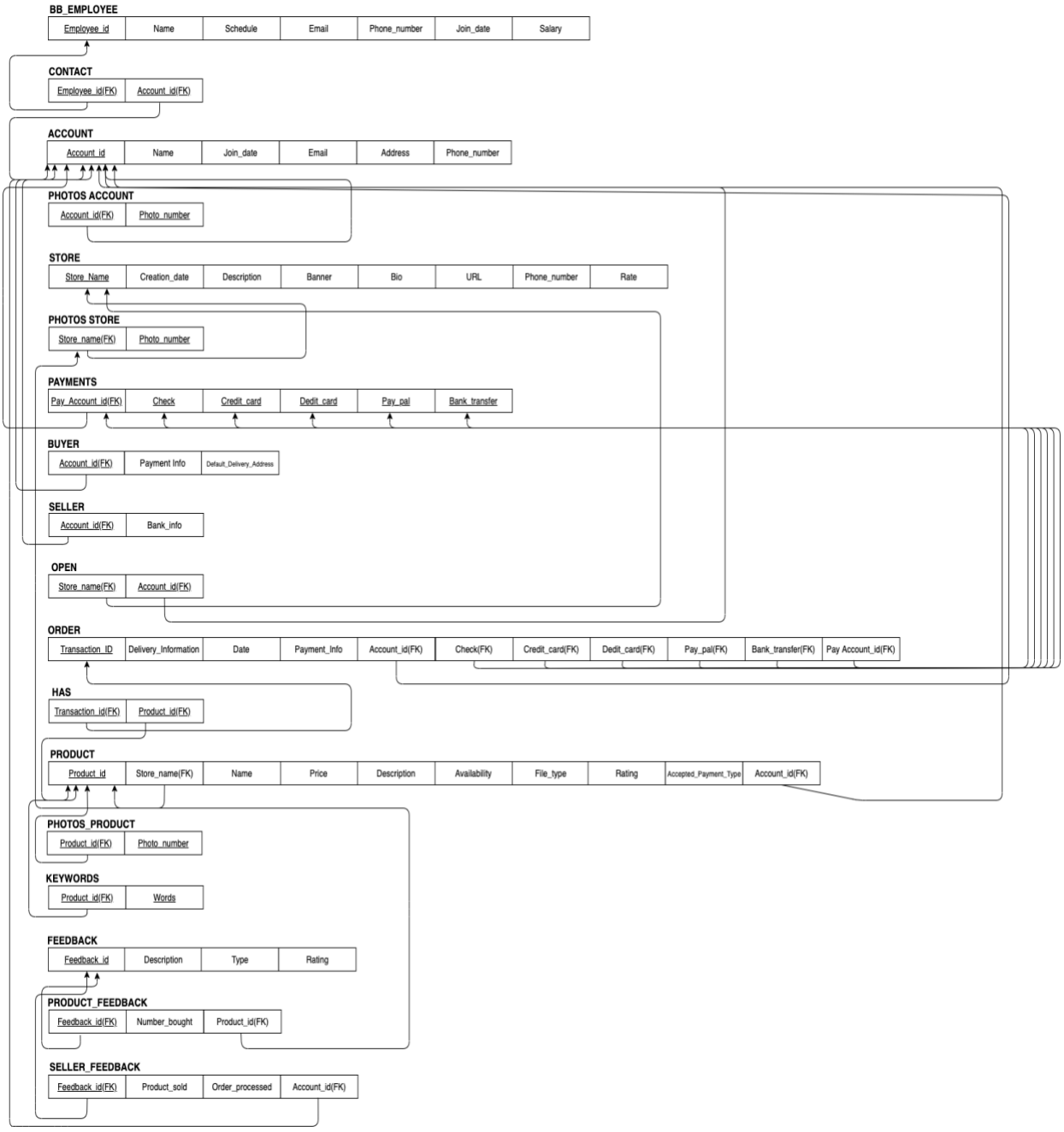
3.1.3 Checkpoint 3

Comments From Grader:

- Each account should have multiple payment methods.
- Payment should be an entity instead of the attribute, and then do not need payment info attributes in other entities; payment of the account should not be an attribute; instead, it should be an entity associated with each account; Also, there should be a relation between order and payment

1. Provide a current version of your ER Diagram and Relational Model as per Project Checkpoint 02. If you were instructed to change the model for Project Checkpoint 02, make sure you use the revised versions of your models


Salary
Join_Date
Emplyee_ID
Email
Schedule
Phone Number
Name
BB_EMPLOYEE
N
CONTACT
M
Photo_number
Name
Join Date
Account_ID
E-Mail
Address
Phone Number
Photo
Type
ACCOUNT
Type
d
'Buyer'
'Seller'
Default Delivery Add
BUYER
Karma Val
Bank Info
SELLER
M
associate
1
Credit Card
Debit Card
PayPal
Check
Bank Transfer
PAYMENT
1
PROVIDES
N
1
1
USED
N
M
OPEN
N
1
CREATE
N
1
LEAVE
1
Rating
Feedback_ID
Description
Type
FEEDBACK
1
ASSOCIATE Product
N
Rate
Phone Number
Creation Date
Store_name
Url
Description
Banner
Bio
Photo_store
Photo_number
STORE
Transaction ID
Date
Delivery Information
ORDER
Type
d
'Product'
'Seller'
M
HAS
N
N
ASSOCIATE Product
1
PRODUCT_FEEDBACK
SELLER_FEEDBACK
Number bought
Products sold
Orders processed
1
Accepted Payment Type
Name
Product ID
Price
Description
File_type
Title
Availability
Photo_product
Photo_number
Rating
Words
Keyword
PRODUCT
N
SELL



3. Simple queries

/* a. Find the titles of all IP Items by a given Seller that cost less than $10 (you choose how to designate the seller, seller is given by Account_id='S0001') */

```
SELECT p.Name
FROM PRODUCT AS p
WHERE p.Account_id = 'S0001' AND p.Price < 10;
```

```
/* b. Give all the titles and their dates of purchase made by given buyer (you choose how to designate the
buyer, buyer is given by Account_id='B0001') */
SELECT p.Name, o.Date_of_order
FROM PRODUCT AS p, ORDER_ AS o, HAS AS h
WHERE h.Transaction_id = o.Transaction_id AND o.Account_id = 'B0001' AND h.Product_id =
p.Product_id;

/* c. Find the seller names for all sellers with less than 5 IP Items for sale  FIXED???*/
SELECT a.Name
FROM PRODUCT AS p, ACCOUNT AS a, SELLER AS s
WHERE s.Account_id = a.Account_id AND p.Account_id = a.Account_id
GROUP BY s.Account_id
HAVING count(p.Product_id) < 5;

/* d. Give all the buyers who purchased a IP Item by a given seller and the names of the IP Items they
purchased, seller given by Account_id = 'S0003' */
SELECT b.*, p.Name
FROM PRODUCT AS p, BUYER AS b, ORDER_ AS o, HAS AS h, ACCOUNT AS a
WHERE p.Account_id = 'S0003' AND  h.Transaction_id = o.Transaction_id AND o.Account_id = 'B0001'
AND h.Product_id = p.Product_id AND b.Account_id = a.Account_id;

/* e. Find the total number of IP Items purchased by a single buyer (you choose how to designate the
buyer, buyer given by account_id='B0004') */
SELECT COUNT(p.Product_id)
FROM PRODUCT AS p, ORDER_ AS o, HAS AS h
WHERE h.Transaction_id = o.Transaction_id AND o.Account_id = 'B0003' AND h.Product_id =
p.Product_id;


/* f. Find the buyer who has purchased the most IP Items and the total number of IP Items they have
purchased */
(1)
        CREATE VIEW BUYER_BUYS_CAL(Name, Number_purchase)
        AS SELECT A.Name,    COUNT(*)
        FROM ACCOUNT AS A, ORDER_ AS ORD, HAS AS HA, PRODUCT AS PR
        WHERE A.Account_id = ORD.Account_id AND ORD.Transaction_id = HA.Transaction_id AND
        HA.Product_id = PR.Product_id
        GROUP BY A.Name;

        SELECT name, number_purchase
        FROM BUYER_BUYS_CAL
        WHERE number_purchase = (
                                SELECT max(number_purchase)
                                FROM BUYER_BUYS_CAL);

(2)
        SELECT A.Name,    COUNT(*)
        FROM ACCOUNT AS A, ORDER_ AS ORD, HAS AS HA, PRODUCT AS PR
```

```
WHERE A.Account_id = ORD.Account_id AND ORD.Transaction_id = HA.Transaction_id AND
HA.Product_id = PR.Product_id
GROUP BY A.Name
HAVING COUNT(*)=
        (SELECT MAX(BC)
        FROM (SELECT   COUNT(*) AS BC
        FROM ACCOUNT AS A, ORDER_ AS ORD, HAS AS HA, PRODUCT AS PR
         WHERE A.Account_id = ORD.Account_id AND ORD.Transaction_id =
HA.Transaction_id AND HA.Product_id = PR.Product_id
        GROUP BY A.Name) );
```

4. Extra queries

```
/*a. List all the Sellers (Seller_id) and the number of stores for each seller
        SELECT      Account_id, COUNT(*)
        FROM        OPEN
        GROUP BY    Account_id;
/*b. Count the number of unique products that are sold in a store called “Super Duper”
        SELECT      COUNT(p.Product_id)
        FROM        PRODUCT AS p
        WHERE       p.Store_name = ‘Super Duper’;
/*c. Count the number of BB_EMPLOYEE whose name is “John”
        SELECT      COUNT(b.Employee_id)
        FROM        BB_EMPLOYEE AS b
        WHERE       b.Name LIKE ‘John%’;
```

5. Advanced queries

```
/*a. Provide a list of buyer names, along with the total dollar amount each buyer has spent.
        SELECT      A.Name, SUM (P.Price)
        FROM        ACCOUNT AS A, BUYER AS B, ORDER_ AS O, HAS AS H, PRODUCT
AS P
        WHERE       A.Account_id=B.Account_id AND B.Account_id=O.Account_id
                    AND O.Transaction_id = H.Transaction_id AND
P.Product_id=H.Product_id

        GROUP BY    A.Name;
```

b. Provide a list of buyer names and e-mail addresses for buyers who have spent more than the average buyer.

```
SELECT      A.Name, A.Email
FROM        ACCOUNT AS A, BUYER AS B, ORDER_ AS O, HAS AS H, PRODUCT AS P
WHERE       A.Account_id=B.Account_id AND B.Account_id=O.Account_id AND
O.Transaction_id = H.Transaction_id AND P.Product_id=H.Product_id
GROUP BY    A.Name
```

```
HAVING SUM(price)>(SELECT AVG(price)
                   FROM ACCOUNT AS A, BUYER AS B, ORDER_ AS O,
HAS AS H, PRODUCT AS P
                   WHERE A.Account_id=B.Account_id AND
B.Account_id=O.Account_id B.Account_id=O.Account_idAND O.Transaction_id =
H.Transaction_id AND P.Product_id=H.Product_id);
```

c. Provide a list of the IP Item names and associated total copies sold to all buyers, sorted from the IP Item that has sold the most individual copies to the IP Item that has sold the least.

d. Provide a list of the IP Item names and associated dollar totals for copies sold to all buyers, sorted from the IP Item that has sold the highest dollar amount to the IP Item that has sold the smallest.

e. Find the most popular Seller (i.e. the one who has sold the most IP Items)

f. Find the most profitable seller (i.e. the one who has brought in the most money)

g. Provide a list of buyer names for buyers who purchased anything listed by the most profitable Seller.

h. Provide the list of sellers who listed the IP Items purchased by the buyers who have spent more than the average buyer.

3.1.4 Checkpoint 4

2.
BB_EMPLOYEE
{Employee_id, email} -> {name, schedule, join_date, salary, phone_number}

CONTACT
KEY{Employee_id, Account_id}

ACCOUNT
{Account_id, email} -> {Name, join_date, address, Phone_number}

PHOTOS_ACCOUNT
KEY{Account_id, Photo_number}

STORE
{Store_name} -> {Creation_date, Description, Banner, Bio, Rate, URL, Phone_number}

PHOTOS_STORE
KEY{Store_name, Photo_number}

PAYMENTS
{Payment_id}->{ Check, Credit_card, Debit_card, Pay_pal, Bank_transfer}

BUYER
{Account_id} -> {Payment_Info, Default_Delivery_Address}

SELLER
{Account_id} -> {Bank_info}

OPEN
KEY{Store_name, Account_id}

ORDER
{Transaction_id}->{Delivery information, date, Account_id, Payment_id}

HAS
{Transaction_id, Product_id} -> {QTY}

PRODUCT
{Product_id} ->{Store_name, Name, Price, Description, Availability, file_type, rating, payment_type, account_id}

PHOTOS_PRODUCT

KEY{Product_id, phone_number}

KEYWORDS
KEY{Product_id, words}

FEEDBACK
{Feedback_id}->{description, type, rating}

PRODUCT_FEEDBACK
{Product_id, Feedback_id}->{number of bought}

SELLER_FEEDBACK
{Feedback_id, Account_id}->{Product_sold, Order_processed}

3.
BB_EMPLOYEE: 1NF
BB_E1:BCNF
{Employee_id, email} -> {name, schedule, join_date, salary}
BB_E2:BCNF
{email}->{phone_number}

CONTACT: BCNF
ACCOUNT: 1NF
ACC1:BCNF
{Account_id, email} -> {Name, join_date, address}
ACC2:BCNF
{email}->{phone_number}

PHOTOS_ACCOUNT: BCNF
STORE: BCNF
PHOTOS_STORE: BCNF
PAYMENTS: BCNF
BUYER: BCNF
SELLER: BCNF
OPEN: BCNF
ORDER: BCNF
HAS: BCNF
PRODUCT: BCNF
PHOTOS_PRODUCT: BCNF
KEYWORDS: BCNF
FEEDBACK: BCNF
PRODUCT_FEEDBACK: BCNF
SELLER_FEEDBACK: BCNF

4. Since all the normal forms are in Boyce-Codd, this question could be skipped.

5.

a.

List all the products, and the total of each sold in descending order.

```
CREATE VIEW
AS SELECT   Product_id, SUM(QTY)
FROM        (PRODUCT JOIN HAS ON Product_id=Product_id)
GROUP BY    Product_id
ORDER BY    SUM(QTY) DESC;
```

B.

List all the sellers and the number of unique products that they sell in descending order

```
CREATE VIEW
AS SELECT   Account_id, sum(Product_id)
FROM        (SELLER JOIN PRODUCT ON Account_id = Account_id)
GROUP BY    Account_id
ORDER BY    SUM(Product_id) DESC;
```

6.

Hash: For hash index, we choose the words attribute from the KEYWORD schema since this attribute will be used when customers want to search the item by keywords. Which will be an equality test.

B-tree: We would index the price attribute of a PRODUCT as a B-tree since this attribute will be used in many range tests. For example, if a user wants to search for products within a certain price range.

7.

a. Insert a new employee into the bb_employee schedule, insert a new store into Open and its information in Store.

```
BEGIN TRANSACTION;
    INSERT OR ROLLBACK INTO BB_EMPLOYEE VALUES('E0021', “Misha Hofstatard”',
    'Schedule', 'mh111@gmail.com', 6147977875, '2018-06-06', 45000);
    INSERT OR ROLLBACK INTO OPEN VALUES (‘Uni’, S0002);
    INSERT OR ROLLBACK INTO STORE VALUES ('Uni', '2017-12-12', NULL, NULL,
    NULL, NULL, 6148302364, 4);
 COMMIT;
```

b. Insert a new product to Product table, also update the default delivery address of Buyer that has Account_id = ‘B0003’ to ‘127 Crest Rd, Columbus, OH'

```
BEGIN TRANSACTION;
```

INSERT OR ROLL BACK INTO PRODUCT VALUES('P0021', 'thdke', '1001010', 1000, NULL, 1, '.bin', NULL, 'S0009');

UPDATE OR ROLLBACK BUYER
SET Default_Delivery_Address = ‘127 Crest Rd, Columbus, OH'
WHERE Account_id=’B0003’;
COMMIT;

3.2 SQLs

3.2.1 CREATE

```
CREATE TABLE BB_EMPLOYEE1
(Employee_id          VARCHAR(15)      NOT NULL,
 Name       VARCHAR(15)      NOT NULL,
Schedule   VARCHAR(15)      NOT NULL,
Email            VARCHAR(100)     NOT NULL,
Join_date        DATE       NOT NULL,
Salary                INT   NOT NULL,
PRIMARY KEY(Employee_id    ));

CREATE TABLE BB_EMPLOYEE2
(Email           VARCHAR(100)     NOT NULL,
Phone_number     INT        ,
FOREIGN KEY (Email) REFERENCES BB_EMPLOYEE1(Email)
ON DELETE SET NULL    ON UPDATE CASCADE);


CREATE TABLE CONTACT
(Employee_id          VARCHAR(15)      NOT NULL,
Account_id       VARCHAR(15)      NOT NULL,
FOREIGN KEY(Employee_id    ) REFERENCES BB_EMPLOYEE1(Employee_id),
FOREIGN KEY(Account_id) REFERENCES ACCOUNT1(Account_id));

CREATE TABLE ACCOUNT1
(Account_id           VARCHAR(15)      NOT NULL,
 Name            VARCHAR(15)      NOT NULL,
Join_date             DATE       NOT NULL,
Email                 VARCHAR(100)     NOT NULL,
Address          VARCHAR(100)     ,
PRIMARY KEY(Account_id));

CREATE TABLE ACCOUNT2
(Email                VARCHAR(100)     NOT NULL,
Phone_number     INT        ,
```

```
FOREIGN KEY(Email) REFERENCES ACCOUNT1(Email)
ON DELETE CASCADE     ON UPDATE CASCADE);



CREATE TABLE PHOTOS_ACCOUNT
(Account_id           VARCHAR(15)      NOT NULL,
Photo_number     INT   NOT NULL,
PRIMARY KEY(Account_id, Photo_number),
FOREIGN KEY(Account_id) REFERENCES ACCOUNT1(Account_id)
ON DELETE CASCADE     ON UPDATE CASCADE);

CREATE TABLE STORE
(Store_name           VARCHAR(30)      NOT NULL,
Creation_date         DATE       NOT NULL,
Description           VARCHAR(100),
Banner                VARCHAR(30),
Bio              VARCHAR(100),
URL              VARCHAR(50),
Phone            INT,
Rate             INT,
PRIMARY KEY(Store_name));

CREATE TABLE PHOTOS_STORE
(Store_name           VARCHAR(30)      NOT NULL,
 Photo_number    INT              NOT NULL,
PRIMARY KEY (Store_name, Photo_number),
FOREIGN KEY(Store_name) REFERENCES STORE(Store_name)
ON DELETE CASCADE     ON UPDATE CASCADE);

CREATE TABLE PAYMENTS
(Payment_id           INT   NOT NULL,
Bank_check       INT,
Credit_card           INT,
Debit_card       INT,
Paypal                INT,
Bank_transfer         INT,
Payment_type          VARCHAR(15),
Bitcoin               INT,
Karma            INT,
Account_id       VARCHAR(15)      NOT NULL,
PRIMARY KEY (Payment_id),
FOREIGN KEY (Account_id) REFERENCES ACCOUNT1(Account_id)
ON DELETE CASCADE     ON UPDATE CASCADE);
```

```
CREATE TABLE BUYER
(Account_id           VARCHAR(15)           NOT NULL,
Payment_info          VARCHAR(100),
Default_Delivery_Address         VARCHAR(100),
Karma      INT
     CHECK(KARMA>-1),
FOREIGN KEY (Account_id) REFERENCES ACCOUNT1(Account_id)
ON DELETE CASCADE     ON UPDATE CASCADE);

CREATE TABLE SELLER
(Account_id           VARCHAR(15)           NOT NULL,
Bank_info       VARCHAR(100),
FOREIGN KEY (Account_id) REFERENCES ACCOUNT1(Account_id)
ON DELETE CASCADE     ON UPDATE CASCADE);


CREATE TABLE OPEN
(Store_name           VARCHAR(30)      NOT NULL,
Account_id       VARCHAR(15)      NOT NULL,
FOREIGN KEY(Store_name)  REFERENCES STORE(Store_name)
ON DELETE CASCADE     ON UPDATE CASCADE,
FOREIGN KEY(Account_id)    REFERENCES ACCOUNT1(Account_id)
ON DELETE CASCADE     ON UPDATE CASCADE);

CREATE TABLE ORDER_
(Transaction_id  VARCHAR(20)      NOT NULL,
Delivery_info         VARCHAR(20),
Date_of_order              DATE,
Account_id            VARCHAR(15)      NOT NULL,
Payment_id       VARCHAR(15)      NOT NULL,
PRIMARY KEY(Transaction_id),
FOREIGN KEY(Account_id)  REFERENCES ACCOUNT1(Account_id)
ON DELETE CASCADE     ON UPDATE CASCADE,
FOREIGN KEY(Payment_id)  REFERENCES PAYMENTS(Payment_id)
ON DELETE CASCADE     ON UPDATE CASCADE);

CREATE TABLE HAS
(Transaction_id  VARCHAR(20)      NOT NULL,
Product_id       VARCHAR(20)      NOT NULL,
Qty              INT              NOT NULL
     CHECK(Qty >0),
FOREIGN KEY(Transaction_id)  REFERENCES ORDER_(Transaction_id)
ON DELETE CASCADE     ON UPDATE CASCADE,
FOREIGN KEY(Product_id)  REFERENCES PRODUCT(Product_id)
```

```
ON DELETE CASCADE     ON UPDATE CASCADE);


CREATE TABLE PRODUCT
(Product_id           VARCHAR(20)      NOT NULL,
Store_name       VARCHAR(30)      NOT NULL,
Name             VARCHAR(20)      NOT NULL,
Price            INT
           CHECK(Price > 0),
Description           VARCHAR(100),
Availability          INT
           CHECK(Availability>-1),
File_type        VARCHAR(20),
Rating                INT,
Accepted_payment_type VARCHAR(15)      NOT NULL,
Account_id            VARCHAR(15)      NOT NULL,
PRIMARY KEY(Product_id),
FOREIGN KEY(Store_name)  REFERENCES STORE(Store_name)
ON DELETE CASCADE     ON UPDATE CASCADE,
FOREIGN KEY(Account_id)  REFERENCES ACCOUNT1(Account_id)
ON DELETE CASCADE     ON UPDATE CASCADE);



CREATE TABLE PHOTOS_PRODUCT
(Product_id           VARCHAR(10)      NOT NULL,
 Photo_number    INT              NOT NULL,
PRIMARY KEY (Product_id, Photo_number),
FOREIGN KEY (Product_id) REFERENCES PRODUCT(Product_id)
     ON DELETE CASCADE     ON UPDATE CASCADE
);

CREATE TABLE KEYWORDS
(Product_id           VARCHAR(10)      NOT NULL,
 Words                VARCHAR(20)      NOT NULL,
PRIMARY KEY (Product_id, Words),
FOREIGN KEY (Product_id) REFERENCES PRODUCT(Product_id)
     ON DELETE CASCADE     ON UPDATE CASCADE
);

CREATE TABLE FEEDBACK
(Feedback_id          VARCHAR(15)      NOT NULL,
 Description          VARCHAR(200)     ,
 Rating               INT              NOT NULL
```

```
            CHECK(Rating>-1 AND Rating <6),
Transaction_id        VARCHAR(15)      NOT NULL,

PRIMARY KEY (Feedback_id)
FOREIGN KEY (Transaction_id) REFERENCES ORDER_(Transaction_id)
     ON DELETE CASCADE     ON UPDATE CASCADE
);
```

3.2.2 INSERT

```
INSERT INTO BB_EMPLOYEE1
VALUES
     ('E0001', 'Michael Scott',  'Schedule', 'scott1@gmail.com',
'2015-12-12', 70000),
     ('E0002', 'Dwight Schrute',  'Schedule', 'schrute09@gmail.com',
'2017-04-12', 60000),
     ('E0003', 'Jim Halpert',  'Schedule', 'halpert123@gmail.com',
'2017-06-01', 60000),
     ('E0004', 'Pam Beasley',  'Schedule', 'pampam@gmail.com',  '2017-
08-04', 50000),
     ('E0005', 'Andy Bernard', 'Schedule', 'bernard23@gmail.com',
'2018-01-12', 55000),
     ('E0006', 'Richard Korf', 'Schedule', 'korh13@gmail.com', '2018-
05-15', 85000),
     ('E0007', 'John Kim', 'Schedule', 'kim938@gmail.com', '2019-10-
18', 70000),
     ('E0008', 'Joseph Kay', 'Schedule', 'kay81@gmail.com', '2017-08-
03', 55000),
     ('E0009', 'Allen Klinger', 'Schedule', 'klinger05@gmail.com',
'2017-06-23', 90000),
     ('E0010', 'Paul Meyer', 'Schedule', 'meyer83@gmail.com', '2019-
01-29', 60000),
     ('E0011', 'James Johnson', 'Schedule', 'jj111@gmail.com', '2019-
01-23', 45000),
     ('E0012', 'Jake Jackson', 'Schedule', 'jj121@gmail.com', '2020-
02-21', 80000),
     ('E0013', 'Tom Smith', 'Schedule', 'ts33@gmail.com', '2017-05-
30', 55000),
     ('E0014', 'Greg Brown', 'Schedule', 'gb1256@gmail.com', '2019-09-
30', 70000),
     ('E0015', 'Tim Simpson', 'Schedule', 'ts6543@gmail.com', '2018-
04-23', 100000),
```

```
      ('E0016', 'Jerry Jackson', 'Schedule', 'jj123@gmail.com', '2020-
05-05', 50000),
      ('E0017', 'Jarrod Newsted', 'Schedule', 'jn123@gmail.com', '2016-
05-30', 51000),
      ('E0018', 'Jena Jackson', 'Schedule', 'jo113@gmail.com',  '2017-
05-05', 53000),
      ('E0019', 'James Hatfield', 'Schedule', 'js144@gmail.com',
'2019-05-05', 48000),
      ('E0020', 'Jerry Schemit', 'Schedule', 'js155@gmail.com', '2018-
05-05', 46000);

INSERT INTO BB_EMPLOYEE2
VALUES
      ('scott1@gmail.com',  2162343212),
      ('schrute09@gmail.com', 6149003400),
      ('halpert123@gmail.com', 6145264993),
      ('pampam@gmail.com', 6145263934),
      ('bernard23@gmail.com', 6145279430),
      ('korh13@gmail.com', 6145264934),
      ('kim938@gmail.com', 6145628042),
      ('kay81@gmail.com', 7346351634),
      ('klinger05@gmail.com', 8273634931),
      ('meyer83@gmail.com', 7236238281),
      ('jj111@gmail.com', 3303743812),
      ('jj121@gmail.com', 6148284932),
      ('ts33@gmail.com', 6145262832),
      ('gb1256@gmail.com', 6145261232),
      ('ts6543@gmail.com', 2161827372),
      ('jj123@gmail.com', 4402832312),
      ('jn123@gmail.com', 2164826361),
      ('jo113@gmail.com', 6142536253),
      ('js144@gmail.com', 6147956666),
      ('js155@gmail.com', 6147977775);



INSERT INTO CONTACT
VALUES
      ('E0001', 'B0001'),
      ('E0002', 'B0002'),
      ('E0003', 'S0003'),
      ('E0004', 'B0004'),
      ('E0005', 'S0002'),
      ('E0006', 'B0006'),
      ('E0007', 'S0006'),
```

```
      ('E0008', 'B0007'),
      ('E0009', 'B0008'),
      ('E0010', 'B0010'),
      ('E0011', 'S0001'),
      ('E0012', 'S0002'),
      ('E0013', 'S0003'),
      ('E0014', 'S0004'),
      ('E0015', 'S0005'),
      ('E0016', 'S0006'),
      ('E0017', 'S0007'),
      ('E0018', 'S0008'),
      ('E0019', 'S0009'),
      ('E0020', 'S0010');


INSERT INTO ACCOUNT1
VALUES
      ('B0001', 'David Wallace', '2015-12-12', 'wallace1@gmail.com',
'67 Euclid Ave, Cleveland,OH'),
      ('B0002', 'Jon Nguyen', '2018-08-04','nguyen1@gmail.com', '424
Overlook, San Francisco,CA'),
      ('B0003', 'Mary Adcock', '2017-06-01', 'adcockt123@gmail.com',
'231 Pope st, Athens, GA'),
      ('B0004', 'Harrison Yi', '2017-08-04',  'yi1@gmail.com', '3422
May Ave, Boston,MA'),
      ('B0005', 'Yoona Im', '2018-01-12', 'im123@gmail.com','78 Cedar
Rd, San Antonio,TX'),
      ('B0006', 'Jack Lim', '2019-04-21', 'lim023@gmail.com', '603
Harley Dr, Columbus, OH'),
      ('B0007', 'David Wang', '2020-10-19', 'wang92@gmail.com', '110
Tibet Rd, Columbus, OH'),
      ('B0008', 'Brian Davis', '2019-03-20', 'davis3@gmail.com', '117 E
Weber Rd, Columbus, OH'),
      ('B0009', 'Dante Simonetti', '2018-05-14', 'simon192@gmail.com',
'128 Crestview Rd, Columbus, OH'),
      ('B0010', 'Robert Mori', '2019-03-20', 'mori999@gmail.com', '72 E
Tulane Rd, Columbus, OH'),
      ('S0001', 'Cade Sbrocco', '2016-07-08', 'sbroccocade@gmail.com',
'5983 Mystic Rdg, Erie, PA'),
      ('S0002', 'Nick Jones', '2018-11-16', 'nwjones@gmail.com', '324
Forbes Ave, Pittsburgh, PA'),
      ('S0003', 'Cory Branton', '2017-05-25', 'cbranton7@gmail.com',
'231 5th Ave, Pittsburgh, PA'),
      ('S0004', 'Luca Lavezzo', '2016-03-12', 'llavez@gmail.com', '423
State St, Cincinnati, OH'),
```

```
      ('S0005', 'Brandon Mannley', '2016-11-15', 'bman@gmail.com', '231
Euclid Ave, Cleveland, OH'),
      ('S0006', 'Ozzy Osburne', '2016-11-15', 'ozzy@gmail.com', '254
Euclid Ave, Cleveland, OH'),
      ('S0007','Steve Harris', '2016-11-15', 'ss@gmail.com', '531
Euclid Ave, Cleveland, OH'),
      ('S0008','Sid Vicious', '2016-11-15', 'sd@gmail.com', '666 Euclid
Ave, Cleveland, OH'),
      ('S0009','Johnny Rotten', '2016-11-15', 'jr@gmail.com', '789
Harley Dr, Columbus, OH'),
      ('S0010','Jeff Hannyman', '2016-11-15', 'jh666@gmail.com', '555
Lane Ave, Cleveland, OH');

INSERT INTO ACCOUNT2
VALUES
      ('wallace1@gmail.com', 6142847278),
      ('nguyen1@gmail.com', 3124675746),
      ('adcockt123@gmail.com',2162343212),
      ('yi1@gmail.com', 61473848383),
      ('im123@gmail.com', 4064738576),
      ( 'lim023@gmail.com', 6149322812),
      ('wang92@gmail.com',6143823723),
      ('davis3@gmail.com', 6143820221),
      ('simon192@gmail.com', 6140392810),
      ('mori999@gmail.com', 6149302812),
      ('sbroccocade@gmail.com', 8149202721),
      ('nwjones@gmail.com', 8142329090),
      ('cbranton7@gmail.com', 8143235400),
      ('llavez@gmail.com', 2341220343),
      ('bman@gmail.com', 9124506783),
      ('ozzy@gmail.com', 6147957783),
      ('ss@gmail.com', 6147958787),
      ('sd@gmail.com', 6147958866),
      ('jr@gmail.com', 6147957979),
      ('jh666@gmail.com', 6147957414);

INSERT INTO PHOTOS_ACCOUNT
VALUES
      ('B0001', 123),
      ('B0002', 124),
      ('B0003', 125),
      ('B0004', 126),
      ('B0005', 127),
      ('B0006', 128),
      ('B0007', 129),
```

```
      ('B0008', 130),
      ('B0009', 131),
      ('B0010', 132),
      ('S0001', 133),
      ('S0002', 134),
      ('S0003', 235),
      ('S0004', 336),
      ('S0005', 437),
      ('S0006', 538),
      ('S0007', 939),
      ('S0008', 340),
      ('S0009', 341),
      ('S0010', 642);


INSERT INTO STORE
VALUES
      ('Go Shop', '2018-01-20', NULL, NULL, NULL, NULL, 6142738273, 4),
      ('Luxx', '2017-05-23', NULL,NULL,NULL,NULL, 6142842842, 5),
      ('Nava', '2019-09-20', NULL,NULL,NULL,NULL, 6141192288, 3),
      ('FaciArt', '2020-05-04', NULL, NULL, NULL,NULL, 6142902008, 4),
      ('Ama Accessory', '2018-12-08', NULL, NULL, NULL,NULL,
6147262008, 5),
      ('Happy Drive', '2016-12-12', NULL, NULL, NULL, NULL, 6148301820,
5),
      ('ABC Mart', '2018-11-03', NULL, NULL, NULL, NULL, 6143812910,
3),
      ('GCG Gaming', '2020-3-25', NULL, NULL, NULL, NULL, 6149301572,
5),
      ('Sweet Spot', '2017-5-29', NULL, NULL, NULL, NULL, 6147292932,
4),
      ('Holy Moly', '2019-10-01', NULL, NULL, NULL, NULL, 6148832933,
4),
      ('Super Duper', '2017-09-18', NULL, NULL, NULL, NULL, 8147789200,
4),
      ('Party Place', '2017-03-08', NULL, NULL, NULL, NULL, 8148882227,
3),
      ('Great Deals', '2019-10-31', NULL, NULL, NULL, NULL, 8143239456,
5),
      ('Stuff 4 U', '2017-12-01', NULL, NULL, NULL, NULL, 8143332290,
3),
      ('Here 2 Help', '2018-02-08', NULL, NULL, NULL, NULL, 8146578902,
2),
      ('Holy Diver', '2017-06-06', NULL, NULL, NULL, NULL, 6147897777,
4),
```

```
      ('Iron Maiden', '2019-02-14', NULL, NULL, NULL, NULL, 6146666666,
3),
      ('Highway 2 Hell', '2018-02-14', NULL, NULL, NULL, NULL,
814666666, 5),
      ('Die Die My Darling', '2017-02-14', NULL, NULL, NULL, NULL,
8145554444, 3),
      ('Dead Rose', '2017-02-14', NULL, NULL, NULL, NULL, 8148888888,
2);


INSERT INTO PHOTOS_STORE
VALUES
      ('Go Shop', 3423),
      ('Luxx', 4324),
      ('FaciArt', 7452),
      ('FaciArt', 47234),
      ('Ama Accessory', 4832),
      ('Happy Drive', 19292),
      ('ABC Mart', 2911),
      ('GCG Gaming', 5838),
      ('Sweet Spot', 39973),
      ('Holy Moly', 22283),
      ('Super Duper', 30192),
      ('Party Place', 32412),
      ('Stuff 4 U',11111),
      ('Great Deals', 32111),
      ('Here 2 Help', 93241),
      ('Holy Diver',00001),
      ('Iron Maiden',00001),
      ('Highway 2 Hell',00001),
      ('Die Die My Darling',00001),
      ('Dead Rose',00001);


      INSERT INTO BUYER
      VALUES
            ('B0001',NULL,'67 Euclid Ave, Cleveland,OH', 100),
            ('B0002',NULL,'424 Overlook, San Francisco,CA', 200),
            ('B0003',NULL,'231 Pope st, Athens, GA', 106),
            ('B0004',NULL,'3422 May Ave, Boston,MA', 55),
            ('B0005',NULL,'78 Cedar Rd, San Antonio,TX', 66),
            ('B0006',NULL, '603 Harley Dr, Columbus, OH', 78),
            ('B0007',NULL, '110 Tibet Rd, Columbus, OH', 79),
            ('B0008',NULL, '117 E Weber Rd, Columbus, OH', 85),
            ('B0009',NULL, '128 Crestview Rd, Columbus, OH', 60),
```

```
('B0010',NULL, '72 E Tulane Rd, Columbus, OH', 100);


INSERT INTO SELLER
VALUES
      ('S0001', NULL),
      ('S0002', 4039293828389232),
      ('S0003', 2839392339232913),
      ('S0004', 3923392349133904),
      ('S0005', 8937918339183813),
      ('S0006', 1939411118392220),
      ('S0007', 1939402018392220),
      ('S0009', 1922391004927281),
      ('S0010', NULL);


INSERT INTO OPEN
VALUES
      ('Go Shop', 'S0001'),
      ('Luxx', 'S0002'),
      ('Nava', 'S0003'),
      ('FaciArt', 'S0004'),
      ('Ama Accessory', 'S0005'),
      ('Happy Drive', 'S0006'),
      ('ABC Mart', 'S0007'),
      ('GCG Gaming', 'S0008'),
      ('Sweet Spot', 'S0009'),
      ('Holy Moly', 'S00010'),
      ('Super Duper', 'S0010'),
      ('Party Place', 'S0003'),
      ('Stuff 4 U', 'S0002'),
      ('Great Deals', 'S0001'),
      ('Here 2 Help', 'S0009'),
      ('Holy Diver', 'S0009'),
      ('Iron Maiden', 'S0006'),
      ('Highway 2 Hell', 'S0005'),
      ('Die Die My Darling', 'S0004'),
      ('Dead Rose', 'S0001');



INSERT INTO PAYMENTS
VALUES
```

```
      (1, NULL, 4004103013933813,NULL,NULL,NULL,'Credit card', NULL,
NULL, 'B0001'),
      (2, NULL, NULL, 3847283238423843, NULL, NULL, 'Debit card', NULL,
NULL,'B0002'),
      (3, NULL,2374274228322831, NULL, NULL, NULL, 'Credit card', NULL,
NULL, 'B0003'),
      (4, NULL,2374274228322831, NULL, NULL, NULL,'Credit card', NULL,
NULL, 'B0004'),
      (5, NULL,2374274228322831, NULL, NULL, NULL,'Credit card', NULL,
NULL, 'B0005'),
      (6, NULL, NULL, NULL, NULL, 21928371314,'Bank transfer', NULL,
NULL, 'B0006'),
      (7, NULL, NULL, 4117774029313922, NULL, NULL, 'Debit card', NULL,
NULL,'B0007'),
      (8, NULL, NULL, 1929472222915974, NULL, NULL, 'Debit card', NULL,
NULL,'B0008'),
      (9, 129482716292, NULL, NULL, NULL, NULL, 'Bank check', NULL,
NULL,'B0009'),
      (10, NULL, NULL, NULL, NULL, 12293928720, 'Bank transfer', NULL,
NULL, 'B0010'),
      (11, 129482713292, NULL, NULL, NULL, NULL, 'Bank check', NULL,
NULL, 'B0001'),
      (12, NULL, NULL, NULL, NULL, 12293928720,'Bank transfer', NULL,
NULL,'B0002'),
      (13, NULL, NULL, 4147774069313922, NULL, NULL,'Debit card', NULL,
NULL, 'B0003'),
      (14, NULL, NULL, 1929332222915974, NULL, NULL,'Debit card', NULL,
NULL, 'B0004'),
      (15, NULL, NULL, NULL, NULL, 66666663720,'Bank transfer', NULL,
NULL, 'B0005'),
      (16,NULL, NULL, NULL, NULL, 21928376666,'Bank transfer', NULL,
NULL, 'B0006'),
      (17,NULL, NULL, NULL, NULL, 2293778720,'Bank transfer', NULL,
NULL, 'B0007'),
      (18,NULL, NULL, NULL, NULL, 2293924440,'Bank transfer', NULL,
NULL, 'B0008'),
      (19,NULL, NULL, NULL, NULL, 2277728720,'Bank transfer', NULL,
NULL, 'B0009'),
      (20,NULL, NULL, NULL, NULL, 2266668720,'Bank transfer', NULL,
NULL, 'B0010');



INSERT INTO ORDER_
```

```
VALUES

      ('T0001', NULL, '2018-10-29','B0001', 1),
      ('T0002', NULL, '2018-12-02', 'B0002', 2),
      ('T0003', NULL, '2019-04-24', 'B0002', 2),
      ('T0004', NULL, '2019-10-09', 'B0001', 11),
      ('T0005', NULL, '2020-10-29', 'B0003', 3),
      ('T0006', NULL, '2020-10-29', 'B0006', 6),
      ('T0007', NULL, '2020-05-23', 'B0003', 13),
      ('T0008', NULL, '2020-10-15', 'B0003', 13),
      ('T0009', NULL, '2019-07-10', 'B0009', 9),
      ('T0010', NULL, '2020-10-01', 'B0010', 20),
      ('T0011', NULL, '2020-11-29', 'B0006',16),
      ('T0012', NULL, '2020-04-23', 'B0007', 7),
      ('T0013', NULL, '2020-11-15', 'B0008', 18),
      ('T0014', NULL, '2019-03-10', 'B0009', 9),
      ('T0015', NULL, '2020-08-01', 'B0010', 10),
      ('T0016', NULL, '2020-10-19', 'B0004', 14),
      ('T0017', NULL, '2020-05-16', 'B0009', 19),
      ('T0018', NULL, '2020-09-20', 'B0010',10),
      ('T0019', NULL, '2019-03-11', 'B0003', 3),
      ('T0020', NULL, '2020-01-01', 'B0007',17);



INSERT INTO HAS
VALUES
      ('T0001', 'P0003',12),
      ('T0002', 'P0005', 10),
      ('T0003', 'P0001', 31),
      ('T0004', 'P0002', 20),
      ('T0005', 'P0003', 40),
      ('T0006', 'P0006', 55),
      ('T0007', 'P0009', 47),
      ('T0008', 'P0010', 48),
      ('T0009', 'P0007', 87),
      ('T0010', 'P0008', 33),
      ('T0011', 'P0011', 30),
      ('T0012', 'P0019', 2),
      ('T0012', 'P0002', 18),
      ('T0013', 'P0013', 34),
      ('T0013', 'P0001', 1),
      ('T0014', 'P0002', 32),
      ('T0014', 'P0020', 24),
      ('T0015', 'P0005', 32),
```

```
      ('T0016', 'P0002', 60),
      ('T0017', 'P0003', 1),
      ('T0018', 'P0009', 20),
      ('T0019', 'P0010', 17),
      ('T0020', 'P0012', 49);


INSERT INTO PRODUCT
VALUES
      ('P0001', 'Go Shop', 'je3',20, NULL, 200, 'txt', 4, 'all',
'S0001'),
      ('P0002', 'Nava', 'rep',4, NULL, 500, 'txt', 2, 'all', 'S0004'),
      ('P0003', 'Luxx', 'plant',10, NULL, 100, 'txt', 4, 'all',
'S0002'),
      ('P0004', 'Go Shop', 'snack',40, NULL, 800, 'txt', 5, 'all',
'S0001'),
      ('P0005', 'Ama Accessory', 'vi',12, NULL, 500, 'txt', 3, 'all',
'S0005'),
      ('P0006', 'Super Duper', 'new_type', 100, NULL, 500, '.pdf', 5,
'all', 'S0010'),
      ('P0007', 'Party Place', '3d_rty', 5, NULL, 1000, '.png',
4,'all', 'S0003'),
      ('P0008', 'Stuff 4 U', 'Jk1234', 20, NULL, 15, '.mp3', 3, 'all',
'S0002'),
      ('P0009', 'Great Deals', 'gr8_file', 450, NULL, 0, '.txt.', 5,
'all', 'S0001'),
      ('P0010', 'Here 2 Help', '1001010', 1000, NULL, 1, '.bin',
5,'all', 'S0009'),
      ('P0011', 'Holy Diver', 'DIO', 666, NULL, 0, '.txt.', 5, 'all',
'S0009'),
      ('P0012', 'Holy Diver', 'Computer as God', 777, NULL, 0, '.txt.',
5, 'all', 'S0009'),
      ('P0013', 'Die Die My Darling', 'Misfits', 1000, NULL, 0,
'.txt.', 4, 'all', 'S0004'),
      ('P0014', 'Iron Maiden', 'Fear of the dark', 789, NULL, 0,
'.bin', 3, 'all', 'S0006'),
      ('P0015', 'Highway 2 Hell', 'AC-DC', 1010, 'Go Dying', 0, '.bin',
2, 'all', 'S0005'),
      ('P0016', 'Happy Drive', 'handle_93', 100, NULL, 15, '.txt', 4,
'all', 'S0007'),
      ('P0017', 'ABC Mart', 'abcsh01', 50, NULL, 500, '.txt', 4, 'all',
'S0002'),
      ('P0018', 'GCG Gaming', 'sc2_012',60, NULL, 100, '.txt',4, 'all',
'S0003'),
```

```
      ('P0019', 'Sweet Spot', 't_1032', 10, NULL, 10, '.txt',2, 'all',
'S0007'),
      ('P0020', 'Holy Moly', 'hm_39', 400, NULL, 0, '.txt', 3,'all',
'S0010');



INSERT INTO PHOTOS_PRODUCT
VALUES
      ('P0001', 02012),
      ('P0002', 00002),
      ('P0003', 00293),
      ('P0004', 01290),
      ('P0005', 03710),
      ('P0006', 00456),
      ('P0007', 00457),
      ('P0008', 00458),
      ('P0009', 00459),
      ('P0010', 00470),
      ('P0011', 00001),
      ('P0012', 00002),
      ('P0013', 00012),
      ('P0014', 00022),
      ('P0015', 01300),
      ('P0016', 03812),
      ('P0017', 03813),
      ('P0018', 03814),
      ('P0019', 03815),
      ('P0020', 03816);


INSERT INTO KEYWORDS
VALUES
      ('P0001', 'ahd'),
      ('P0002', 'ha'),
      ('P0003', 'den'),
      ('P0004', 'sweet'),
      ('P0005','hat'),
      ('P0006', 'X'),
      ('P0007', 'helpful'),
      ('P0008', 'useful'),
      ('P0009', 'needed'),
      ('P0010', 'new'),
      ('P0011', 'dio'),
      ('P0012', 'black sabbath'),
```

```
      ('P0013', 'punk'),
      ('P0014', 'piece of mind'),
      ('P0015','metal'),
      ('P0016', 'alloy'),
      ('P0017', 'leather'),
      ('P0018', 'game'),
      ('P0019', 'sweet'),
      ('P0020', 'holy');



INSERT INTO FEEDBACK
VALUES
      ('F0001', NULL, 4,'T0001'),
      ('F0002', NULL,  4,'T0002'),
      ('F0003', NULL, 5, 'T0003'),
      ('F0004', NULL, 2, 'T0004'),
      ('F0005', NULL, 4,'T0005'),
      ('F0006', 'Nice Item', 4,'T0006'),
      ('F0007', 'Worth the money', 4,'T0007'),
      ('F0008', 'Great seller', 5,'T0008'),
      ('F0009', NULL, 2, 'T0009'),
      ('F0010', NULL,4, 'T0010'),
      ('F0011', 'I do not like it', 2, 'T0011'),
      ('F0012', 'This is so cool', 5,'T0012'),
      ('F0013', 'Customer service is really bad', 1, 'T0013'),
      ('F0014', 'Not bad at all', 3, 'T0014'),
      ('F0015', 'Seller is so kind', 5, 'T0015'),
      ('F0016','Not bad at all', 4, 'T0016'),
      ('F0017', 'Not bad at all', 4, 'T0017'),
      ('F0018', 'Not bad at all',4, 'T0018'),
      ('F0019', 'Not bad at all', 3, 'T0019'),
      ('F0020', 'Very good', 5, 'T0020');
```

3.2.3 VIEW

```
CREATE VIEW PRODUCT_QTY AS
SELECT     P.Product_id, SUM(H.QTY)
FROM       (PRODUCT AS P JOIN HAS AS H ON P.Product_id=H.Product_id)
GROUP BY   P.Product_id
ORDER BY   SUM(H.QTY) DESC;
```

```
CREATE VIEW     SELLER_PRODUCT  AS
SELECT     S.Account_id, COUNT(P.Product_id)
FROM       (SELLER AS S JOIN PRODUCT AS P ON S.Account_id =
P.Account_id)
GROUP BY   S.Account_id
ORDER BY   COUNT(P.Product_id) DESC;
```

3.2.4 INDEX

```
CREATE INDEX price_index
ON PRODUCT(price);


CREATE INDEX keyword_index
ON KEYWORDS(words);
```

3.2.5 QUERIES

```
/* (3a) Find the titles of all IP Items by a given Seller that cost
less than $10
(you choose how to designate the seller, seller is given by
Account_id='S0001') */
SELECT     p.Name
FROM       PRODUCT AS p
WHERE      p.Account_id = 'S0001' AND p.Price < 10;

/* (3b) Give all the titles and their dates of purchase made by given
buyer (you choose how to designate the buyer, buyer is given by
Account_id='B0001') */
SELECT     p.Name, o.Date_of_order
FROM       PRODUCT AS p, ORDER_ AS o, HAS AS h
WHERE      h.Transaction_id = o.Transaction_id AND o.Account_id =
           'B0001' AND h.Product_id = p.Product_id;

/* (3c) Find the seller names for all sellers with less than 5 IP
Items for sale */
SELECT     a.Name
FROM       PRODUCT AS p, ACCOUNT1 AS a, SELLER AS s
WHERE      s.Account_id = a.Account_id AND p.Account_id = a.Account_id
GROUP BY   s.Account_id
HAVING     count(p.Product_id) < 5;
```

```
/* (3d) Give all the buyers who purchased a IP Item by a given seller
and the names of the IP Items they purchased
Seller designated by Account_id = S_id */
SELECT     a.Name,P.Name
FROM       PRODUCT AS p, ORDER_ AS o, HAS AS h, ACCOUNT1 AS a
WHERE      p.Account_id = 'S0003' AND  h.Transaction_id =
           o.Transaction_id AND o.Account_id = a.Account_id AND
           h.Product_id = p.Product_id;


/* (3e) Find the total number of IP Items purchased by a single buyer
(you choose how to designate the buyer)
Buyer designated by Account_id = B_id */
SELECT     COUNT(p.Product_id)
FROM       PRODUCT AS p, ORDER_ AS o, HAS AS h
WHERE      h.Transaction_id = o.Transaction_id AND o.Account_id =
           'B0003' AND h.Product_id = p.Product_id;

/* (3f) Find the buyer who has purchased the most IP Items and the
total number of IP Items they have purchased */
SELECT     A.Name, COUNT(*)
FROM       ACCOUNT1 AS A, ORDER_ AS ORD, HAS AS HA, PRODUCT AS PR
WHERE      A.Account_id = ORD.Account_id AND ORD.Transaction_id =
           HA.Transaction_id AND HA.Product_id = PR.Product_id
GROUP BY   A.Name
HAVING     COUNT(*)=
                (SELECT MAX(BC)
                FROM (SELECT   COUNT(*) AS BC
                     FROM       ACCOUNT1 AS A, ORDER_ AS ORD, HAS AS
                                HA, PRODUCT AS PR
                     WHERE      A.Account_id = ORD.Account_id AND
                                ORD.Transaction_id = HA.Transaction_id
                                AND HA.Product_id = PR.Product_id
                     GROUP BY   A.Name));

/* (4a) List all the Sellers (Seller_id) and the number of stores for
each seller*/
SELECT     Account_id, COUNT(*)
FROM       OPEN
GROUP BY   Account_id;

/* (4b) Count the number of unique products that are sold in a store
called “Super Duper”*/
SELECT     COUNT(p.Product_id)
```

```
FROM       PRODUCT AS p
WHERE      p.Store_name = 'Super Duper'

/* (4c) Count the number of BB_EMPLOYEE whose name is "John" */
SELECT     COUNT(b.Employee_id)
FROM       BB_EMPLOYEE1 as b
WHERE      b.Name LIKE 'John%';

/* (5a) Provide a list of buyer names, along with the total dollar
amount each buyer has spent. */
SELECT     A.Name, SUM (P.Price*H.Qty)
FROM       ACCOUNT1 AS A, BUYER AS B, ORDER_ AS O, HAS AS H, PRODUCT
           AS P
WHERE      A.Account_id=B.Account_id AND B.Account_id=O.Account_id
           AND O.Transaction_id = H.Transaction_id AND
           P.Product_id=H.Product_id
GROUP BY   A.Name;

/* (5b) Provide a list of buyer names and e-mail addresses for buyers
who have spent more than the average buyer.*/
SELECT     A.Name,A.Email
FROM       ACCOUNT1 AS A, BUYER AS B, ORDER_ AS O, HAS AS H, PRODUCT
           AS P
WHERE      A.Account_id=B.Account_id AND B.Account_id=O.Account_id AND
           O.Transaction_id = H.Transaction_id AND
           P.Product_id=H.Product_id
GROUP BY   A.account_id
HAVING     SUM(H.Qty*P.Price)>(SELECT
                      SUM(H.Qty*P.Price)/COUNT(DISTINCT A.Account_id)
                      FROM       ACCOUNT1 AS A, BUYER AS B, ORDER_ AS
                                 O, HAS AS H, PRODUCT AS P
                      WHERE      A.Account_id=B.Account_id AND
                                 B.Account_id=O.Account_id AND
                                 O.Transaction_id = H.Transaction_id
                                 AND   P.Product_id=H.Product_id);

/* (5c) Provide a list of the IP Item names and associated total
copies sold to all buyers,
 sorted from the IP Item that has sold the most individual copies to
the IP Item that has sold the least.*/
SELECT     PR.Name, SUM(HA.QTY)
FROM       ACCOUNT1 AS A, ORDER_ AS ORD, HAS AS HA, PRODUCT AS PR
WHERE      A.Account_id = ORD.Account_id AND ORD.Transaction_id =
           HA.Transaction_id AND HA.Product_id = PR.Product_id
GROUP BY   PR.Product_id
```

```
ORDER BY   SUM(HA.QTY) DESC

/* (5d) Provide a list of the IP Item names and associated dollar
totals for copies sold to all buyers,
sorted from the IP Item that has sold the highest dollar amount to the
IP Item that has sold the smallest.*/
SELECT      P.name,SUM(qty)*P.price AS total
FROM        PRODUCT AS P, HAS AS H
WHERE       P.Product_id=H.Product_id
GROUP BY    P.product_id
ORDER BY    SUM(H.qty)*P.price DESC;

/* (5e) Find the most popular Seller (i.e. the one who has sold the
most IP Items) */
SELECT     A2.Name
FROM       ACCOUNT1 AS A1, ORDER_ AS ORD1, HAS AS HA1, PRODUCT AS PR1,
           ACCOUNT1 AS A2
WHERE      A1.Account_id = ORD1.Account_id AND ORD1.Transaction_id =
           HA1.Transaction_id AND HA1.Product_id = PR1.Product_id AND
           PR1.Account_id = A2.Account_id
GROUP BY   PR1.Account_id
HAVING     SUM(HA1.QTY)=
        (SELECT       MAX(QT)
        FROM          (SELECT    SUM(HA.QTY) AS QT
                      FROM       ACCOUNT1 AS A, ORDER_ AS ORD, HAS AS
                                 HA, PRODUCT AS PR
                      WHERE      A.Account_id = ORD.Account_id AND
                                 ORD.Transaction_id = HA.Transaction_id
                                 AND HA.Product_id = PR.Product_id
                      GROUP BY   PR.Account_id));

/* (5f) Find the most profitable seller (i.e. the one who has brought
in the most money) */
SELECT     A2.Name
FROM       ACCOUNT1 AS A1, ORDER_ AS ORD1, HAS AS HA1, PRODUCT AS PR1,
           ACCOUNT1 AS A2
WHERE      A1.Account_id = ORD1.Account_id AND ORD1.Transaction_id =
           HA1.Transaction_id AND HA1.Product_id = PR1.Product_id AND
           PR1.Account_id = A2.Account_id
GROUP BY   PR1.Account_id
HAVING     SUM(HA1.QTY*PR1.Price)=
           (SELECT MAX(QT)
                FROM (SELECT SUM(HA.QTY*PR.Price) AS QT
                      FROM  ACCOUNT1 AS A, ORDER_ AS ORD, HAS AS HA,
                            PRODUCT AS PR
```

```
                WHERE      A.Account_id = ORD.Account_id AND
                           ORD.Transaction_id = HA.Transaction_id AND
                           HA.Product_id = PR.Product_id
             GROUP BY      PR.Account_id));

/* (5g) Provide a list of buyer names for buyers who purchased
anything listed by the most profitable Seller.*/
SELECT     a.name
FROM       ACCOUNT1 AS a, BUYER AS b, HAS AS h, ORDER_ AS O, (SELECT
           PR1.Account_id, PR1.Product_id

           FROM       ACCOUNT1 AS A1, ORDER_ AS ORD1, HAS AS HA1,
                      PRODUCT AS PR1

           WHERE      A1.Account_id = ORD1.Account_id AND
                      ORD1.Transaction_id = HA1.Transaction_id AND
                      HA1.Product_id = PR1.Product_id

           GROUP BY   PR1.Account_id

           HAVING     SUM(HA1.QTY*PR1.Price)=

                      (SELECT    MAX(QT)

                      FROM (SELECT SUM(HA.QTY*PR.Price) AS QT

                      FROM  ACCOUNT1 AS A, ORDER_ AS ORD, HAS AS HA,
                            PRODUCT AS PR

                      WHERE A.Account_id = ORD.Account_id AND
                            ORD.Transaction_id = HA.Transaction_id AND
                            HA.Product_id = PR.Product_id

                      GROUP BY PR.Account_id)) ) AS p


WHERE      p.Product_id = h.Product_id AND o.Transaction_id =
           h.Transaction_id AND o.Account_id = b.Account_id AND
           b.Account_id = a.Account_id;

/*(5h) Provide the list of sellers who listed the IP Items purchased
by the buyers who have spent more than the average buyer.*/
SELECT     A1.Name
FROM       ACCOUNT1 AS A1, PRODUCT AS p, SELLER AS s, HAS AS h, ORDER_
AS O, (SELECT    B.*
```

```
            FROM        ACCOUNT1 AS A, BUYER AS B, ORDER_ AS O, HAS AS H,
                        PRODUCT AS P
            WHERE       A.Account_id=B.Account_id AND
                        B.Account_id=O.Account_id AND O.Transaction_id =
                        H.Transaction_id AND  P.Product_id=H.Product_id
           GROUP BY     A.account_id
            HAVING      SUM(H.Qty*P.Price)>(SELECT
                        SUM(H.Qty*P.Price)/COUNT(DISTINCT A.Account_id)
                           FROM     ACCOUNT1 AS A, BUYER AS B,
                                    ORDER_ AS O,HAS AS H, PRODUCT AS P
                           WHERE    A.Account_id=B.Account_id AND
                                    B.Account_id=O.Account_id AND
                                    O.Transaction_id = H.Transaction_id
                                    AND   P.Product_id=H.Product_id)) as b

WHERE       p.Product_id = h.Product_id AND o.Transaction_id =
            h.Transaction_id AND o.Account_id = b.Account_id AND
            p.Account_id = s.Account_id AND s.Account_id =
      A1.Account_id;
```

3.2.6 INSERT/DELETE (New Samples)

```
      INSERT INTO PRODUCT
      VALUES
      ('P0021', 'Go Shop', 'jrr3',20, NULL, 200, 'txt', 4, NULL,
'S0001'),
      ('P0022', 'Nava', 'repo',4, NULL, 500, 'txt', 2, NULL, 'S0004'),
      ('P0023', 'Luxx', 'play',10, NULL, 100, 'txt', 4, NULL, 'S0002'),
      ('P0024', 'Go Shop','stop here',40, NULL, 800, 'txt', 5, NULL,
'S0001'),
      ('P0025', 'Ama Accessory', 'vi',12, NULL, 500, 'txt', 3, NULL,
'S0005');


      INSERT INTO STORE
      VALUES
      ('Gold', '2008-01-20', NULL, NULL, NULL, NULL, 6142738273, 4),
      ('Luxy', '2015-05-23', NULL,NULL,NULL,NULL, 6142842842, 5),
      ('Naria', '2014-09-20', NULL,NULL,NULL,NULL, 6141192288, 3),
      ('Hyper', '2022-05-04', NULL, NULL, NULL,NULL, 6142902008, 4),
      ('Ale Accessory, '2012-12-08', NULL, NULL, NULL,NULL, 6147262008,
5),
```

```
     ('Happy Dive', '2013-12-12', NULL, NULL, NULL, NULL, 6148301820,
5);


     DELETE FROM STORE
     WHERE Store_name = 'Go Shop';

     DELETE FROM PRODUCT
     WHERE Product_id = 'P0001';
```