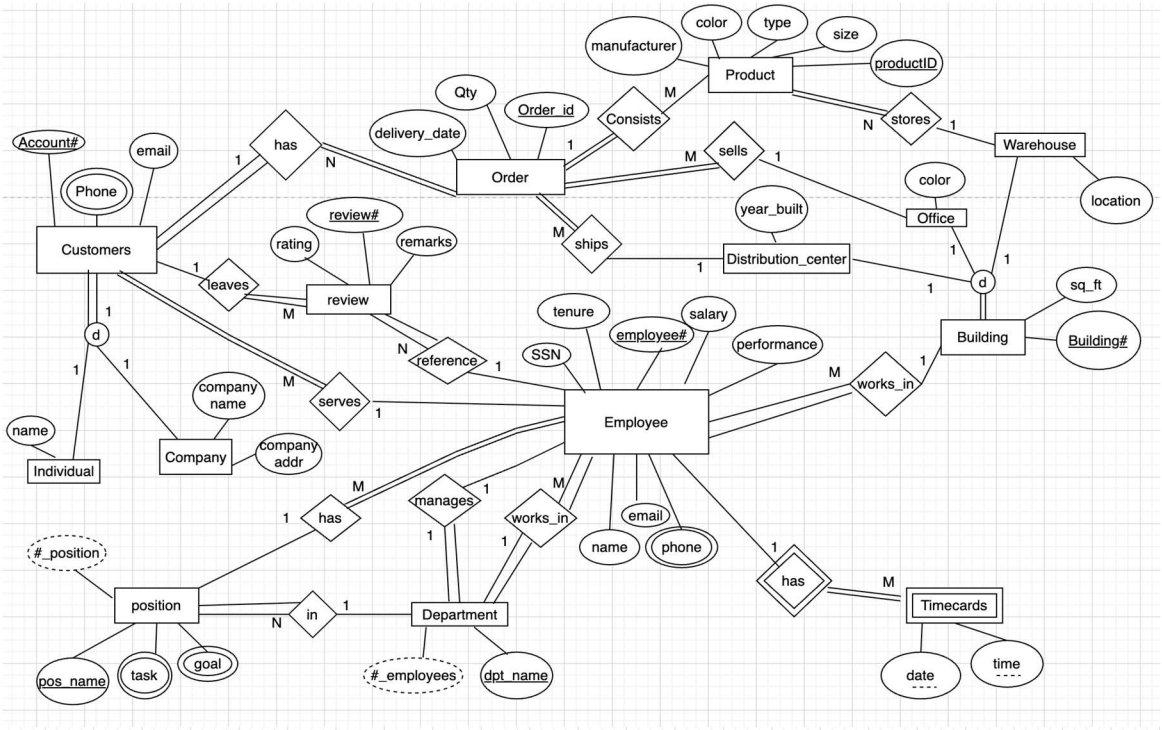
Final Project Report

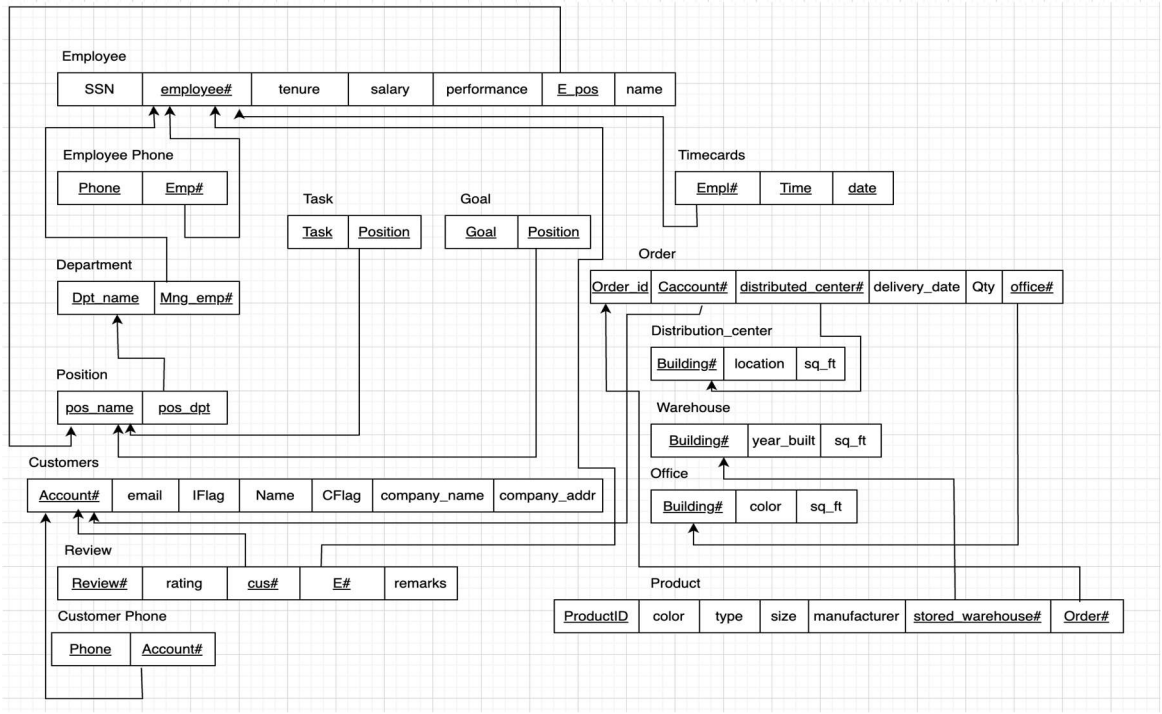
Bradley Burk, Claire Erdelyi, Ian Brown, and Walid Sirag

07/15/25

Part I

EER and Relational Schema





Functional Dependencies and Normalization Levels

Employee(SSN, employee#, tenure, salary, performance, E pos)
{employee#} -> {SSN, tenure, salary, performance, E pos}
The employee entity is in BCNF because employee# is a super key and the FD is in 3NF. SSN does not determine a unique tuple because it is possible for an employee to not have an SSN listed.

Employee Phone(Phone, employee#)
{employee#, Phone} -> {employee#, Phone}
The Employee Phone entity is in BCNF because {employee#, Phone} is a super key and the FD is in 3NF.

Position(pos_name, pos dpt)
{pos_name}-> {pos dpt}
The Position entity is in BCNF because {pos_name} is a super key and the FD is in 3NF.

Customers(Account#, email, IFlag, Name, CFlag, company_name, company_addr)
{Account#} -> {email, IFlag, Name, CFlag, company_name, company_addr}
The Customer entity is in BCNF because Account# is a super key and the FD is in 3NF.

Review(Review#, rating, remarks, cus#, ESSN, Ename)
{Review#} -> {rating, Remarks, cus#, ESSN, Ename}
The Review entity is in BCNF because Review# is a super key and the FD is in 3NF.

Timecards(Empl# ,Time, Date)
{Empl#, Time, Date}-> {Empl#, Time, Date}
The Timecards entity is in BCNF because {Empl#, Time, Date} is a super key and the FD is in 3NF.

Task(Task, Position)
{Task, Position} -> {Task, Position}
The Task entity is in BCNF because {Task, Position} is a super key and the FD is in 3NF.

Goal(Goal, Position)
{Goal, Position} -> {Goal, Position}
The Goal entity is in BCNF because {Goal, Position} is a super key and the FD is in 3NF.

Orders(Order id, Caccount#, distributed center#, office#, delivery_date, Qty)
{Order_id} -> {Caccount#, distributed_center#, office#, delivery_date, qty}
The Orders entity is in BCNF because Order_id is a super key and the FD is in 3NF.

Product(ProductID, color, type, size, manufacturer, stored_warehouse#, Order#)
{ProductID} -> {color, type, size, manufacturer, stored_warehouse#, Order#}
The Product entity is in BCNF because ProductID is a super key and the FD is in 3NF.

Distribution Center(Building#, location, sq. Ft.)

{Building#} -> {location, sq. Ft.}

The Distribution Center entity is in BCNF because Building# is a super key and the FD is in 3NF.

Warehouse(Building#, year_built, sq. Ft.)

{Building#} -> {year_built, sq. Ft.}

The Warehouse entity is in BCNF because Building# is a super key and the FD is in 3NF.

Office(Building#, color, sq. Ft.)

{Building#} -> {color, sq. Ft.}

The Office entity is in BCNF because Building# is a super key and the FD is in 3NF.

Indexes

Description 1: Hash index of size in Product table.

Rational 1: Size is an important characteristic of paper. There are many different sizes of paper, so using a hash index is preferable.

Description 2: Hash index of Tenure in the Employee table.

Rational 2: Tenure is one of the attributes that determines that salary an employee receives. Due to the many different values of tenure present, using a hash index is preferred.

Description 3: Hash index of qty in Orders table

Rational 3: Using a hash index here due to the many different values present. Indexing qty is useful to quickly obtain which orders, and thus products, have the highest and lowest qty rates.

Views

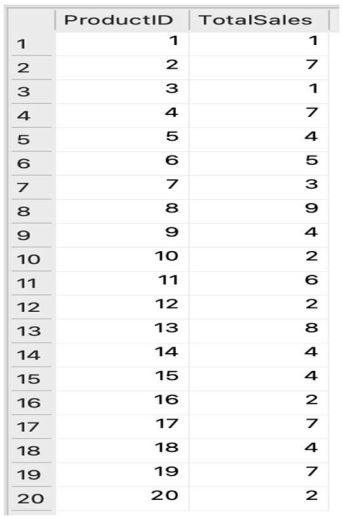
English: View for seeing total number of sales for each product

Relational Algebra:

Result <- π ProductID, qty (ProductID $\mathscr{F}$ SUM qty (Product ⋈ {order_number = Order_no} Orders))

SQL:

```
CREATE VIEW ProductSales AS
    SELECT P.ProductID, SUM(O.qty) AS TotalSales
    FROM Product AS P, Orders AS O
    WHERE P.order_number = O.Order_no
    GROUP BY P.ProductID;
```



	ProductID	TotalSales
1	1	1
2	2	7
3	3	1
4	4	7
5	5	4
6	6	5
7	7	3
8	8	9
9	9	4
10	10	2
11	11	6
12	12	2
13	13	8
14	14	4
15	15	4
16	16	2
17	17	7
18	18	4
19	19	7
20	20	2

English: View for total number of employees in each position

Relational Algebra:

Result <- π count_pos, pos_name (pos_name $\mathscr{F}$COUNT pos (Position ⋈ {pos_name = pos} Employee))

SQL:

```
CREATE VIEW PositionNum AS
    SELECT POS.pos_name, COUNT(E.pos) AS TotalEmployeed
    FROM Position AS POS, Employee AS E
    WHERE POS.pos_name = E.pos
    GROUP BY POS.pos_name
```

	pos_name	TotalEmployeed
1	Acc.Manager	1
2	Account Ex.	2
3	Accountant	2
4	Clerk	1
5	Finance Manager	1
6	Financial Analyst	2
7	Loan Officer	1
8	Op.Analyst	1
9	Op.Coordinator	1
10	P.Agent	1
11	P.Manager	1
12	Project Manager	1
13	RD Manager	1
14	Res.Engineer	1
15	Research Scientist	1
16	Sales Dev.Rep	2
17	Sales Manager	1
18	Tax Accountant	1
19	Treasurer	1

Transactions

```
-- Give employees above a rating a raise
BEGIN TRANSACTION;
UPDATE EMPLOYEE SET SALARY = SALARY + 1000 WHERE Performance >= 5;

SELECT SALARY FROM EMPLOYEE WHERE Performance >= 5;
END TRANSACTION;


--Get names and orders of a certain individual customer
BEGIN TRANSACTION;

SELECT PersonName, cus_account FROM Orders OUTER JOIN Customer ON Order_no =
Account_no
```

```
WHERE cus_account = 6 & CompanyName IS NULL & PersonName IS NOT NULL;

COMMIT;
END TRANSACTION;

-- select reviews less than 5 with name
BEGIN TRANSACTION;

SELECT CompanyName, rating, remarks FROM Review OUTER JOIN Customer ON cus_no
= Account_no WHERE rating < 5 & PersonName IS NULL;

COMMIT;
END TRANSACTION;
```

Section 2 – User Manual

a. Description for each table

The Paper Company database consists of 15 tables: Customer, Customer_phone, Department, Distribution_center, Employee, Goal, Office, Orders, Phone, Position, Product, Review, Task, Timecard, and Warehouse.

The Customer table has 5 columns: Account_no, Email, CompanyName, PersonName, and CompanyAddr. Account_no keeps track of each customer's account number. It is of integer data type and is the primary key for the table. All other attributes are of the variable character data type. Each customer must have an email address associated with an account number. CompanyName may have NULL values if the customer is an individual and not a company; vise-versa for PersonName. CompanyAddr must be included if the customer is a company.

Customer_phone consists of 2 columns: Customer_phone and cust_no. Customer_phone is a variable character data type and cust_no is an integer. Both attributes are the primary key of the table and cust_no is a foreign key associated with the Customer table column Account_no.

The Department table has 2 columns: dept_name and Mng_emp_no. dept_name is a variable character and is the primary key. Mn_emp_no is an integer that is a foreign key referring to the SSN of the employee managing the department.

The Distibution_center table has 3 columns: building_no, location, and sq_ft. Both the building_no and sq_ft are integers, and the location is variable character. This table has no foreign keys. The primary key is building_no.

The Employee table has 9 columns: employee_no, SSN, Name, Email, Tenure, Salary, Performance, and pos. Data types include integer for employee_no, Tenure, Salary, and Performance with all other attributes being variable characters. The table has 1 foreign keys: pos. Pos refers to the table Position, column pos_name and deptName refers to the table Department, column dept_name. The primary key for this table is employee_no.

The Goal table keeps track of a position's goals and includes 2 columns: Goal and postion. Both columns make up the primary key and position is a foreign key referring to pos_name in the Position table. Both columns are variable characters.

The Office table keeps track of office buildings and includes 3 columns: building_no, sq_ft, and color. Like the Distibution_center table, building_no is the primary key and there are no foreign keys. Both building_no and sq_ft are integers, and color is a variable character.

The Orders table documents each order placed by all customers and includes 6 columns: Order_no, cus_account, distibution_no, delivery_date, qty, and office_no. All attributes are integers, aside from delivery_date, which is a variable character. The primary for this table is Order_no. There are 3 foreign keys: cus_account, distibution_no, and office_no. Cus_account refers to the Customer table column Account_no, while distribution_no and office_no refer to the Distibution_center and Office tables, columns building_no respectively.

The Phone table tracks an employee's phone number(s) and has 2 columns: Phone and emp_no. Both columns make up the primary key and emp_no is a foreign key referring to the Employee table column employee_no. Phone is considered a variable character data type and emp_no an integer data type.

The Position table includes all positions within the company and has 2 columns: pos_name and pos_dept, both of which are considered variable character data types. The pos_name column is the primary key of the table and pos_dept is a foreign key referring to the Department table column dept_name.

The Product table consists of 7 different columns: ProductID, color, type, size, warehouse_no, order_number, and Manufacturer. ProductID, size, warehouse_no, and order_number are all considered integers, while color, type, and Manufacturer are all considered variable character data types. The primary key is ProductID and there are 2 foreign keys: warehouse_no and order_number. Warehouse_no refers to the Warehouse table column building_no and order_ number refers to the Orders table column Order_no.

The Review table tracks customer reviews of their experience with an employee and consists of 5 columns: review_no, rating, cus_no, empNumber, and remarks. Remarks are considered variable characters, while all others are considered integer data types. The primary key is review_no and there are 2 foreign keys: cus_no and empNumber, referring to the table Customer column Account_no and table Employee column employee_no respectively.

The Task table records what tasks are associated with what positions and contains 2 columns: Task and position. The primary key consists of both columns and the position column is a foreign key referring to the Position table column pos_name. Both columns are considered a variable character data type.

The Timecard table tracks each employee's time worked and has 3 columns: Time, date, and employeeNumber. All 3 columns make up the primary key and employeeNumber is a foreign key referring to the Employee table column employee_no. Both Time and employeeNumber are an integer data type and date is a variable character data type.

The final table is Warehouse, which tracks warehouse building numbers and has 3 columns: building_no, year-built, and sq_ft. Building_no is the primary key. There are no foreign keys within this table. All columns are considered to be integer data types.

b. Sample SQL Queries
    1. Query 1:
        i. Find all reviews about an Employee with a rating of 3 or higher
        ii. π_(name,remarks) (SELECT_(rating≥3) (REVIEW JOIN_(employee#=E#) EMPLOYEE))
        iii. SELECT name, remarks FROM Employee JOIN Review ON employee_no = empNumber WHERE rating >= 3;
        iv. Tested and correct

v.

	Name	remarks
1	David Welsch	Great
2	Chris Dickson	Happy with it
3	Hailey Redden	Glad I bought it
4	Harry Vallory	Need more now
5	Joy Pratt	So good
6	Nicole Quinnell	Love it
7	Harry Vallory	None
8	Mason Pickard	Need more now
9	Adela Baxter	Love it
10	Noah Ashley	So good
11	Joy Pratt	Nice
12	Harry Vallory	So good
13	David Welsch	None
14	Nicole Quinnell	Great
15	Rick Ross	Happy with it

2. Query 2:
   i. Give all instances of paper that was purchased and provide the delivery dates and Customer account number.
   ii. π_(account#,delivery_date) (Customers JOIN_(account#=Caccount#) Orders
   iii. SELECT Account_no, delivery_date FROM Orders JOIN Customer ON Account_no = cus_account;
   iv. Tested and correct
   v.

	Account_no	delivery_date
1	19	12/29/2024
2	20	6/24/2025
3	19	5/2/2024
4	8	9/5/2023
5	7	7/10/2023
6	3	11/2/2023
7	9	2/1/2025
8	17	8/5/2023
9	9	5/2/2024
10	4	9/19/2024
11	21	5/28/2025
12	12	1/23/2024

3. Query 3:
   i. List all buildings by square footage and their unique building number assigned.
   ii. π_(building#, sq_ft) (Warehouse) UNION π_(building#, sq_ft (Distribution_center) UNION π_(building#, sq_ft) (Office)

iii. SELECT building_no, sq_ft FROM Warehouse UNION SELECT building_no, sq_ft FROM Distribution_center UNION SELECT building_no, sq_ft FROM Office

iv. Tested and correct

v.

	building_no	sq_ft
1	1	9077
2	1	15832
3	2	7887
4	2	15550
5	3	6371
6	3	14288
7	4	8297
8	4	9310
9	5	8342
10	5	10538
11	6	10104
12	6	15911
13	7	5953
14	7	10012
15	8	13136
16	8	14055
17	9	11769
18	9	13061
19	10	9742
20	10	15775
21	11	12587
22	11	15751
23	12	8915
24	12	11779
25	13	6761
26	13	7562
27	14	5732
28	14	13985
29	15	7835
30	15	15735
31	16	8847
32	16	15436
33	17	8094
34	17	11990
35	18	10665
36	18	15515
37	19	6714
38	19	13350
39	20	10256
40	20	13179
41	21	10102
42	22	7331
43	23	7971
44	24	8750
45	25	8352
46	26	13078
47	27	15558
48	28	7886
49	29	5578
50	30	8948
51	31	15622
52	32	6281
53	33	7644
54	34	13375
55	35	12864
56	36	10074
57	37	7194
58	38	14230
59	39	9411
60	40	8871
61	41	7224
62	42	9243
63	43	12347
64	44	8828

4. Query 4:
   i. Provide the customers information that have received packages from the warehouse in Cleveland.
   ii. π_(account#, email) (Customer JOIN_(account#=Caccount# (SELECT_(location=^' Cleveland^') (Order JOIN_(shippedwarehouse#=building#) Distribution_center)))
   iii. SELECT Account_no, email FROM Orders JOIN Distribution_center ON building_no = distribution_no JOIN Customer ON Account_no = cus_account WHERE location LIKE ‘Cleveland’;
   iv. Tested and correct

	Account_no	Email
1	9	Naomi_Coll513987061@bqkv0.tech

   v.

5. Query 5:
   i. Find the total unique customer accounts that have purchased paper from the company.
   ii. F_(COUNT account#) (Customers)
   iii. SELECT count(Account_no) FROM Customer;
   iv. Tested and correct

	count(Account_no)
1	24

   v.

6. Query 6:
    i. Find the customer account that has purchased the most amount of paper.
    ii. F_(Max count_orderid) (Account# F_(COUNT orderid) (Customers JOIN_(account#=Caccount#) Order))
    iii. SELECT Account_no, count(Order_no) as numberOfOrders FROM Order JOIN Customer ON Account_no = cus_account GROUP BY Account_no ORDER BY numberOfOrders desc LIMIT 1;
    iv. Tested and correct
    v.

	Account_no	numberOfOrders
1	9	3

7. Extra Query 1:
    i. Get the phone numbers of all customers that purchased at least 5 pieces of paper from the company.
    ii. Pi Phone (Order {Caccount = Account, Qty >= 5} (Customer * Phone))
    iii. SELECT - Phone FROM – (Orders JOIN Customer ON cus_account = Account_no) JOIN Phone ON Account_no = emp_no WHERE – Qty >= 5
    iv. Tested and correct
    v.

	Phone
1	1-331-548-1075
2	3-127-135-2353
3	2-456-141-0616
4	6-605-765-6822
5	3-228-365-4255
6	2-456-141-0616
7	6-785-711-3526
8	8-377-767-2372

8. Extra Query 2:
    i. Get the number of employees that clocked in between 6 and 7 on their timecard per department.
    ii. Dpt name F Count (Department JOIN {Dpt name = E dpt} (Employee JOIN {ESSN = SSN, Time > 40} Timecards)
    iii. SELECT – COUNT(SSN), dept_name FROM – (Employee JOIN Timecard ON employee_no = employeeNumber JOIN Department ON deptName = dept_name WHERE – Time LIKE ‘6%’ GROUP BY – dept_name;
    iv. Tested and correct

	COUNT(SSN)	dept_name
1	1	Accounting
2	1	Purchasing
3	1	Research and Development

v.

9. Extra Query 3
    i. Every rating less than or equal to 2 for every employee in a department.
    ii. Pi rating (Department JOIN {Dept name = E dpt, rating <= 2} (Employee JOIN {ESSN = SSN} Review))
    iii. SELECT – Name, rating, dept_name FROM – (Review JOIN Employee ON employee_no=empNumber) JOIN Department ON dept_name= deptName WHERE – rating <= 2 ORDER BY – dept_name
    iv. Tested and correct

	Name	rating	dept_name
1	Robyn Clayton	1	Operations
2	Kendra Kidd	2	Purchasing
3	Martin Hepburn	1	Purchasing
4	Piper Bentley	2	Purchasing
5	Luke Hudson	1	Sales

    v.

10. Advanced Query 1:
    i. Find the number of orders placed for each customer that has placed at least one order.
    ii. R1 := σ_{Account_no = cus_account} (Customer ⋈ Orders)
    R2 := π_{Name, cus_account} (R1)
    R3 := γ_{Name; count(cus_account) → NumberOfInteractions} (R2)
    Result := Sort_{NumberOfInteractions descending} (R3)
    iii. SELECT COALESCE(CompanyName, PersonName) AS Name, count(cus_account) AS NumberOfInteractions FROM Customer JOIN Orders ON Account_no = cus_account GROUP BY Name ORDER BY NumberOfInteractions desc;
    iv. Tested and correct

v.

	Name	NumberOfInteractions
1	Vodafone	4
2	Biolife Grup	3
3	Zepter	2
4	Metro Cash&Carry	2
5	Juliette Wright	2
6	Russel Fleming	1
7	Norah Kirby	1
8	Matt Welsch	1
9	Holly Antcliff	1
10	Global Print	1
11	Elijah Coleman	1
12	Coca-Cola Company	1

11. Advanced Query 2:
    i. Find the number of orders placed for each customer that has placed more orders than the average customer.
    ii. R1 := σ_{Account_no = cus_account} (Customer ⋈ Orders)
    R2 := π_{Name, cus_account} (R1)
    R3 := γ_{Name; count(cus_account) → NumberOfInteractions} (R2)
    Sub1 := σ_{Account_no = cus_account} (Customer ⋈ Orders)
    Sub2 := π_{Name, cus_account} (Sub1)
    Sub3 := γ_{Name; count(cus_account) → NumOrders} (Sub2)
    AvgNumOrders := avg(π_{NumOrders}(Sub3))
    R4 := σ_{NumberOfInteractions > AvgNumOrders} (R3)
    Result := Sort_{NumberOfInteractions descending} (R4)
    iii. SELECT COALESCE(CompanyName, PersonName) AS Name, count(cus_account) AS NumberOfInteractions FROM Customer JOIN Orders ON Account_no = cus_account GROUP BY Name HAVING NumberOfInteractions > (SELECT avg(NumOrders) FROM (SELECT COALESCE(CompanyName, PersonName) AS Name, count(cus_account) AS NumOrders FROM Customer JOIN Orders ON Account_no = cus_account GROUP BY Name)) ORDER BY NumberOfInteractions desc;
    iv. Tested and correct

v.

	Name	NumberOfInteractions
1	Vodafone	4
2	Biolife Grup	3
3	Zepter	2
4	Metro Cash&Carry	2
5	Juliette Wright	2

12. Advanced Query 3:
    i. Find all products that have been purchased by at least one customer.
    ii. R1 := σ_{order_number = Order_no} (Product ⋈ Orders)
    R2 := π_{ProductID, type, color, order_number} (R1)
    R3 := γ_{ProductID, type, color; count(order_number) → InteractionCount} (R2)
    Result := Sort_{Order_no} (R3)
    iii. SELECT ProductID, type, color, count(order_number) as InteractionCount FROM Product JOIN Orders ON order_number = Order_no GROUP BY ProductID ORDER BY Order_no;
    iv. Tested and correct
    v.

	ProductID	type	color	InteractionCount
1	1	IJIOy0uZII	Orange	1
2	2	AXwRN58jRw	Rosewood	1
3	3	QjKN1WEjQg	Fuchsia	1
4	4	VxmhCquM3S	Brown	1
5	5	46o1Xr8zjK	Fuchsia	1
6	6	g4nt0oDMjK	Auburn	1
7	7	vfwXRbs2bI	Auburn	1
8	8	aPLwmXmRKz	Azure	1
9	9	EawUvrHn6c	Aquamarine	1
10	10	xM1EwsUrNy	Purple	1
11	11	kHckqNNzXS	White	1
12	12	NRfexIjtjZ	Cerise	1
13	13	0zu0mvhNGA	Silver	1
14	14	z9DZ4v7ZeD	Olive	1
15	15	2NyAsBenXY	Brown	1
16	16	UjKHRUlBTk	Champagne	1
17	17	oInVwiHl2T	coral	1
18	18	J1lbl82PbX	Black	1
19	19	0k9inI0NXV	Olive	1
20	20	gZDbQvB9LR	Lime	1

13. Advanced Query 4:

i. Find the number of hours worked by each employee

ii. R1 := σ_{employee_no = employeeNumber} (Employee ⋈ Timecard)
R2 := π_{employeeNumber, Name, Tenure, Time} (R1)
R3 := γ_{employeeNumber, Name, Tenure; sum(Time) → hoursWorked} (R2)
Result := Sort_{hoursWorked descending} (R3)

iii. SELECT Name, Tenure, sum(Time) as hoursWorked FROM Employee JOIN Timecard ON employee_no = employeeNumber GROUP BY employeeNumber ORDER BY hoursWorked desc;

iv. Tested and correct

v.

	Name	Tenure	hoursWorked
1	Melania Warner	4	23
2	Noah Ashley	16	20
3	Nicole Quinnell	10	20
4	Hailey Redden	6	18
5	Joy Pratt	24	15
6	Harry Vallory	18	13
7	Piper Bentley	24	13
8	Michaela Ellery	16	13
9	Martin Hepburn	28	12
10	Doug Dillon	28	12
11	Robyn Clayton	27	11
12	Chris Dickson	12	10
13	Luke Hudson	23	10
14	Adela Baxter	24	10
15	Fred Emmett	20	9
16	Rick Ross	13	8
17	Erick Owens	9	8
18	Isla Stone	11	6
19	Renee Chester	1	6
20	Danny Garner	8	4
21	Jules Tanner	17	4
22	Kendra Kidd	18	4
23	David Welsch	27	2
24	Mason Pickard	7	1

14. Advanced Query 5:

i. Find the manufacturer that have manufactured products that have been purchased the most by any customer.

ii. R1 := σ_{Order_no = order_number} (Product ⋈ Orders)

R2 := π_{Manufacturer, qty} (R1)
R3 := γ_{Manufacturer; sum(qty) → itemsSold} (R2)
MaxItems := max(π_{itemsSold}(R3))
Result := σ_{itemsSold = MaxItems} (R3)

iii. SELECT Manufacturer, max(itemsSold) FROM (SELECT Manufacturer, sum(qty) as itemsSold FROM Product JOIN Orders ON Order_no = order_number GROUP BY Manufacturer ORDER BY itemsSold desc);

iv. Tested and correct

	Manufacturer	max(itemsSold)
1	Buckeye Paper	26

v.

15. Advanced Query 6:
    i. Find the quantity sold of each product.
    ii. R1 := σ_{order_number = Order_no} (Product ⋈ Orders)
    R2 := π_{ProductID, type, color, qty} (R1)
    R3 := γ_{ProductID, type, color; sum(qty) → Frequency} (R2)
    Result := Sort_{Frequency descending} (R3)
    iii. SELECT ProductID, type, color, sum(qty) AS Frequency FROM Product JOIN Orders ON order_number = Order_no GROUP BY ProductID ORDER BY Frequency desc;
    iv. Tested and correct

	ProductID	type	color	Frequency
1	8	aPLwmXmRKz	Azure	9
2	13	0zu0mvhNGA	Silver	8
3	2	AXwRN58jRw	Rosewood	7
4	4	VxmhCquM3S	Brown	7
5	17	olnVwiHl2T	coral	7
6	19	0k9inl0NXV	Olive	7
7	11	kHckqNNzXS	White	6
8	6	g4nt0oDMjK	Auburn	5
9	5	46o1Xr8zjK	Fuchsia	4
10	9	EawUvrHn6c	Aquamarine	4
11	14	z9DZ4v7ZeD	Olive	4
12	15	2NyAsBenXY	Brown	4
13	18	J1Ibl82PbX	Black	4
14	7	vfwXRbs2bl	Auburn	3
15	10	xM1EwsUrNy	Purple	2
16	12	NRfexIjtjZ	Cerise	2
17	16	UjKHRUlBTk	Champagne	2
18	20	gZDbQvB9LR	Lime	2
19	1	IJIOy0uZIl	Orange	1
20	3	QjKN1WEjQg	Fuchsia	1

v.

16. Advanced Query 7:
    i. Find the contact information of the customer who has purchased the largest quantity of product.
    ii. R1 := σ_{order_number = Order_no} (Product ⋈ Orders)
    R2 := π_{ProductID, cus_account, type, color, qty} (R1)
    R3 := γ_{ProductID, cus_account, type, color; sum(qty) → Frequency} (R2)
    MaxFreq := max(π_{Frequency}(R3))
    R4 := σ_{Frequency = MaxFreq} (R3)
    TopCustomer := π_{cus_account, Frequency} (R4)
    R5 := σ_{cust_no = Account_no} (Customer ⋈ Customer_phone)
    R6 := σ_{Account_no = cus_account} (R5 ⋈ TopCustomer)
    Result := π_{Name, Email, Customer_phone} (R6)
    iii. SELECT COALESCE(CompanyName, PersonName) AS Name, Email, Customer_phone FROM Customer JOIN Customer_phone ON cust_no = Account_no JOIN(SELECT MAX(Frequency), cus_account FROM (SELECT ProductID, cus_account, type, color, sum(qty) AS Frequency FROM Product JOIN Orders ON order_number = Order_no GROUP BY ProductID ORDER BY Frequency desc)) ON Account_no = cus_account;
    iv. Tested and correct

	Name	Email	Customer _phone
1	Elijah Coleman	Elijah_Coleman435141770@ohqqh.center	1-378-262-1264

    v.
17. Advanced Query 8:
    i. List the customers who have placed more orders than the average customer, along with all the manufacturers that they have manufactured the customer has purchased.
    ii. C1 := σ_{Account_no = cus_account} (Customer ⋈ Orders)
    C2 := π_{Name, cus_account, Order_no} (
    ρ_{Name←COALESCE(CompanyName, PersonName)} (C1)
    C3 := γ_{Name; count(cus_account) → NumOrders} (C2)
    AvgOrders := avg(π_{NumOrders}(C3))
    C4 := γ_{Name, Order_no, cus_account; count(cus_account) → NumberOfInteractions} (C2)
    C5 := σ_{NumberOfInteractions > AvgOrders} (C4)
    P1 := σ_{order_number = Order_no} (Product ⋈ C5)
    QualifiedNames := π_{Name} (P1)
    MainJoin1 := σ_{Account_no = cus_account} (Customer ⋈ Orders)

MainJoin2 := σ_{Order_no = order_number} (MainJoin1 ⋈ Product)
MainJoin2Named := ρ_{Name ← COALESCE(CompanyName, PersonName)} (MainJoin2)
MainFiltered := σ_{Name ∈ QualifiedNames} (MainJoin2Named)
Result := γ_{Manufacturer, Name} (MainFiltered)

iii. SELECT COALESCE(CompanyName, PersonName) AS Name, Manufacturer FROM Customer JOIN Orders ON Account_no = cus_account JOIN Product ON Order_no = order_number WHERE Name IN (SELECT Name FROM Product JOIN (SELECT COALESCE(CompanyName, PersonName) AS Name, count(cus_account) AS NumberOfInteractions, Order_no, cus_account FROM Customer JOIN Orders ON Account_no = cus_account GROUP BY Name HAVING NumberOfInteractions > (SELECT avg(NumOrders) FROM (SELECT COALESCE(CompanyName, PersonName) AS Name, count(cus_account) AS NumOrders, Order_no FROM Customer JOIN Orders ON Account_no = cus_account GROUP BY Name)) ORDER BY NumberOfInteractions desc) ON order_number = Order_no) GROUP BY Manufacturer, Name;

iv. Tested and correct

v.

	Name	Manufacturer
1	Biolife Grup	Buckeye Paper
2	Metro Cash&Carry	Buckeye Paper
3	Vodafone	Buckeye Paper
4	Biolife Grup	Crescent Paper
5	Juliette Wright	International Paper
6	Vodafone	International Paper
7	Juliette Wright	Millcraft
8	Zepter	Millcraft
9	Vodafone	Ohio Paper Tube
10	Vodafone	Ohio Pulp Mills
11	Biolife Grup	Roosevelt Paper
12	Metro Cash&Carry	Roosevelt Paper

c. INSERT Statement Syntax

Customer --> INSERT needs to occur before adding new Customer_phone or the customer's review

INSERT INTO Customer

(Account_no, Email, CompanyName, PersonName, CompanyAddr)

VALUES (25, example@gmail.com, Astro, NULL, 'example lane, 440');

Customer_phone --> INSERT after adding a new Customer

INSERT INTO Customer_phone

(Phone, Account#)

VALUES (x-xxx-xxx-xxxx, 25);

Review

INSERT INTO Review

(review_no, rating, cus_no, empNumber, remarks)

VALUES (21, 5, 25, 12, Nice);

Office

INSERT INTO Office

(building_no, sq_ft, color)

VALUES (66, 5520, Green);

Distibution_center

INSERT INTO Distibution_center

(building_no, location, sq_ft)

VALUES (65, ‘example way, 2220’, 7456);

Warehouse

INSERT INTO Warehouse

(building_no, year_built, sq_ft)

VALUES (66, ‘example way, 2288’, 8888);

Orders --> If new customer, INSERT occurs after adding new cust. All buildings need to be updated if an order is coming from a new building

INSERT INTO Orders

(Order_no, cus_account, distribution_no, delivery_date, qty, office_no)

VALUES (20, 4, 23, 8/12/2025, 3, 39);

Product

INSERT INTO Product

(ProductID, color, type, size, warehouse_no, order_number, Manufacturer)

VALUES (21, Orange, xxxxxxx, 25, 39, 66, Buckeye Paper);

Department --> Stand alone, no tables need updated before INSERT

INSERT INTO Department

(dept_name, Mng_emp_no)

VALUES (example, 7);

Position --> If part of new Department, Department needs INSERT first

INSERT INTO Position

(pos_name, pos_dept)

VALUES (L.Accountant, Accounting);

Goal --> Position INSERT first. Can do Goal or Task in any order

INSERT INTO Goal

(Goal, position)

VALUES (Oversee accounting, L.Accountant);

Task --> Position INSERT first. Can do Task or Goal in any order

INSERT INTO Task

(Task, position)

VALUES (Accounting accuracy, L.Accountant);

Employee --> INSERT must happen before adding new timecards for a new employee or a phone for said employee

INSERT INTO Employee

(employee_no, SSN, Name, Email, Tenure, Salary, Performance, pos)

VALUES (25, Bob Dylan, bob_dylan@gmail.com, 15, 8, L.Accountant);

Timecard

INSERT INTO Timecard

(Time, date, employeeNumber)

VALUES (15, 8/5/2024, 25);

Phone

INSERT INTO Phone

(Phone, emp_no)

VALUES (x-xxx-xxx-xxxx, 25);

d. DELETE Statement Syntax

Customer_phone --> DELETE before Customer

DELETE FROM Customer_phone

(Phone, Account#)

WHERE(x-xxx-xxx-xxxx, 25);

Review --> DELETE before Customer. This must be deleted prior to removing an associated employee as well.

DELETE FROM Review

(review_no, rating, cus_no, empNumber, remarks)

WHERE(21, 5, 25, 12, Nice);

Customer --> DELETE needs to occur after removing Customer_phone or the customer's review

DELETE FROM Customer

(Account_no, Email, CompanyName, PersonName, CompanyAddr)

WHERE(25, example@gmail.com, Astro, NULL, 'example lane, 440');

Office --> DELETE occurs after Orders

DELETE FROM Office

(building_no, sq_ft, color)

WHERE(66, 5520, Green);

Distibution_center --> DELETE happens after Orders and Product

DELETE FROM Distibution_center

(building_no, location, sq_ft)

WHERE(65, ‘example way, 2220’, 7456);

Warehouse -> DELETE happens after Product

DELETE FROM Warehouse

(building_no, year_built, sq_ft)

WHERE(66, ‘example way, 2288’, 8888);

Product --> Must be deleted before buildings if trying to delete said buildings.

DELETE FROM Product

(ProductID, color, type, size, warehouse_no, order_number, Manufacturer)

WHERE(21, Orange, xxxxxxx, 25, 39, 66, Buckeye Paper);

Orders --> DELETE occurs after removing Product. All buildings need to be updated prior to deletion if it is the only order coming from that building

DELETE FROM Orders

(Order_no, cus_account, distribution_no, delivery_date, qty, office_no)

WHERE(20, 4, 23, 8/12/2025, 3, 39);

Department --> Goal, Task, and Position must be deleted before this if they are the only ones in a single department.

DELETE FROM Department

(dept_name, Mng_emp_no)

WHERE(example, 7);

Goal --> DELETE before Position. Can do Goal or Task in any order

DELETE FROM Goal

(Goal, position)

WHERE(Oversee accounting, L.Accountant);

Task --> DELETE before Position. Can do Task or Goal in any order

DELETE FROM Task

(Task, position)

WHERE(Accounting accuracy, L.Accountant);

Position --> If this is the only position in a department, this must be updated first.

DELETE FROM Position

(pos_name, pos_dept)

WHERE(L.Accountant, Accounting);

Timecard

DELETE FROM Timecard

(Time, date, employeeNumber)

WHERE(15, 8/5/2024, 25);

Phone

DELETE FROM Phone

(Phone, emp_no)

WHERE(x-xxx-xxx-xxxx, 25);

Employee --> DELETE must happen after deleting timecards or a phone for an employee

DELETE FROM Employee

(employee_no, SSN, Name, Email, Tenure, Salary, Performance, pos)

WHERE(25, Bob Dylan, bob_dylan@gmail.com, 15, 8, L.Accountant);

Part 2

Java project importing instructions:

Take the unzipped project folder and import it into Eclipse using the usual method. Make sure you have the binary version of our database downloaded and you know its file path. When our project is successfully imported, go to our DATABASE String variable at the top of our FinalSubmission.java file and change the file path to where you stored the binary version of our database while keeping in mind the format of the previous path there (make sure the number and way the slashes you put in are right and that after the final slash the exact name of our database is there in the format (dbName.db). Also make sure to add the external JAR that lets SQLite connect with Java. Run the program and the database should connect.

Employee Entity Manipulation:

Adding Employee:

```
The driver name is SQLite JDBC
The connection to the database was successful.
Main Menu:
1. Customers
2. Employees
3. Orders
4. Positions
5. Departments
6. Useful Reports.
Please enter the number of the submenu you would like to go to, or press 0 to quit:
2
Employee Submenu
1. Add new Employee
2. Search Employee
3. Delete Employee
4. Edit Employee
5. Back to Main Menu
1
Enter SSN:
629570489
Enter new employee number:
740
Enter tenure:
1
Enter salary:
100
Enter name:
Jemma
Enter performance rating:
1
Enter email:
JemmaPell@wherever.fr
Enter position of employee:
Treasurer
New employee successfully inserted!
Employee Submenu
1. Add new Employee
2. Search Employee
3. Delete Employee
4. Edit Employee
5. Back to Main Menu
```

Results in SQLite:

23	546738927	Martin Hepburn	Martin_Hepburn1875100187@jh02o.page	28	86776	2	P. Agent
24	987654321	Danny Garner	Danny_Garner756006504@hepmv.website	8	65855	8	Project Manager
740	629570489	Jemma	JemmaPell@wherever.fr	1	100	1	Treasurer

Searching for an Employee:

```
Employee Submenu
1. Add new Employee
2. Search Employee
3. Delete Employee
4. Edit Employee
5. Back to Main Menu
2
Input Employee number (or 'x' to quit):
740
employee_no,  SSN,  Name,  Email,  Tenure,  Salary,  Performance,  pos
740,  629570489,  Jemma,  JemmaPell@wherever.fr,  1,  100,  1,  Treasurer
```

Editing Employee (doing edits in order):

```
Employee Submenu
1. Add new Employee
2. Search Employee
3. Delete Employee
4. Edit Employee
5. Back to Main Menu
4
Enter the number of the emplyee you want to edit:
740
1. Employee Number
2. SSN
3. Name
4. Email
5. Tenure
6. Salary
7. Performance
8. Position
Enter the number of the attribute you want to edit or press 0 to stop:
1
Enter new Employee Number:
741
Employee number changed!

1. Employee Number
2. SSN
3. Name
4. Email
5. Tenure
6. Salary
7. Performance
8. Position
Enter the number of the attribute you want to edit or press 0 to stop:
2
Enter new SNN:
175930728
Employee SSN changed!
```

```
1. Employee Number
2. SSN
3. Name
4. Email
5. Tenure
6. Salary
7. Performance
8. Position
Enter the number of the attribute you want to edit or press 0 to stop:
3
Enter new Name:
Jenna
Employee name changed!

1. Employee Number
2. SSN
3. Name
4. Email
5. Tenure
6. Salary
7. Performance
8. Position
Enter the number of the attribute you want to edit or press 0 to stop:
4
Enter new Email:
JennaPell@wherever.fr
Employee Email changed!

1. Employee Number
2. SSN
3. Name
4. Email
5. Tenure
6. Salary
7. Performance
8. Position
Enter the number of the attribute you want to edit or press 0 to stop:
5
Enter new Tenure:
2
Employee Tenure changed!
```

```
1. Employee Number
2. SSN
3. Name
4. Email
5. Tenure
6. Salary
7. Performance
8. Position
Enter the number of the attribute you want to edit or press 0 to stop:
6
Enter new Salary:
100001
Employee Salary changed!
```

```
1. Employee Number
2. SSN
3. Name
4. Email
5. Tenure
6. Salary
7. Performance
8. Position
Enter the number of the attribute you want to edit or press 0 to stop:
7
Enter new Performance:
5
Employee Performance changed!
```

```
1. Employee Number
2. SSN
3. Name
4. Email
5. Tenure
6. Salary
7. Performance
8. Position
Enter the number of the attribute you want to edit or press 0 to stop:
8
Enter new Position:
Clerk
Employee Position changed!
```

Result of edits in SQLite:

23	546738927	Martin Hepburn	Martin_Hepburn1875100187@jh02o.page	28	86776	2	P. Agent
24	987654321	Danny Garner	Danny_Garner756006504@hepmv.website	8	65855	8	Project Manager
741	175930728	Jenna	JennaPell@wherever.fr	2	100001	5	Clerk

Deleting Employee:

```
Employee Submenu
1. Add new Employee
2. Search Employee
3. Delete Employee
4. Edit Employee
5. Back to Main Menu
3
Enter Employee ID of employee to be deleted:
741
Employee successfully deleted
```

Result of deletion in SQLite:

22	546372819	David Welsch	David_Welsch1320883737@kyb7t.shop	27	136274	3	Clerk
23	546738927	Martin Hepburn	Martin_Hepburn1875100187@jh02o.page	28	86776	2	P. Agent
24	987654321	Danny Garner	Danny_Garner756006504@hepmv.website	8	65855	8	Project Manager

Functions with Orders (for customers):

Adding an order:

```
Main Menu:
1. Customers
2. Employees
3. Orders
4. Positions
5. Departments
6. Useful Reports.
Please enter the number of the submenu you would like to go to, or press 0 to quit:
3
Would you like to search or create an order? 1 for create, 2 for search
1
Enter order details to create order
Enter order num
21
Enter cust account
1
Enter distribution center
1
Enter delivery_date
7/19/25
Enter quantity
3
Enter office
25
Order added
```

Searching for an order and returning relevant information to the customer:

```
Main Menu:
1. Customers
2. Employees
3. Orders
4. Positions
5. Departments
6. Useful Reports.
Please enter the number of the submenu you would like to go to, or press 0 to quit:
3
Would you like to search or create an order? 1 for create, 2 for search
2
Insert Order num
20
Search results:
Order_no,  cus_account,  delivery_date,  qty
20,  3,  7/6/2024,  2
```

Function with Position:

Shows all Employees with entered position:

```
Main Menu:
1. Customers
2. Employees
3. Orders
4. Positions
5. Departments
6. Useful Reports.
Please enter the number of the submenu you would like to go to, or press 0 to quit:
4
Which position?
Accountant
Search results:
pos,  Name,  Salary,  Email,  Performance
Accountant,  Michaela Ellery,  162277,  Michaela_Ellery2119408289@6ijur.audio,  9
Accountant,  Noah Ashley,  120353,  Noah_Ashley291714758@fhuux.website,  2
```

Function with Department:

Shows all employees and their attributes in the department specified:

```
Main Menu:
1. Customers
2. Employees
3. Orders
4. Positions
5. Departments
6. Useful Reports.
Please enter the number of the submenu you would like to go to, or press 0 to quit:
5
Which department?
Sales
Search results:
Name,  pos,  Salary,  Email,  Performance
Adela Baxter,  Sales Manager,  158740,  Adela_Baxter905751416@bcfhs.org,  6
Luke Hudson,  Account Ex.,  130484,  Luke_Hudson255201470@mpibr.edu,  1
Fred Emmett,  Sales Dev. Rep,  43557,  Fred_Emmett769331947@xtwt3.club,  7
Isla Stone ,  Account Ex.,  118642,  Isla_Stone 306832800@nb44i.solutions,  3
Chris Dickson,  Sales Dev. Rep,  170636,  Chris_Dickson1719673659@qu9ml.pro,  6
```

Useful reports :

- Total number of items a user has interacted with. Retrieve the user/customer with the most interactions incorporated.

```
The driver name is SQLite JDBC
The connection to the database was successful.
Main Menu:
1. Customers
2. Employees
3. Orders
4. Positions
5. Departments
6. Etc.
7. Useful Reports.
Please enter the number of the submenu you would like to go to, or press 0 to quit:
7
Useful reports:
1. Total item interactions per customer
2. Most popular item
3. Most used manufacturer
Enter numerical selection or enter 0 to leave
1
Report: Total item interactions per customer
Account_no,  Name,  NumberOfInteractions
9,  Vodafone,  3
3,  Biolife Grup,  3
19,  Juliette Wright,  2
12,  Zepter,  2
7,  Metro Cash&Carry,  2
22,  Russel Fleming,  1
21,  Holly Antcliff,  1
20,  Matt Welsch,  1
17,  Elijah Coleman,  1
13,  Norah Kirby,  1
8,  Vodafone,  1
4,  Coca-Cola Company,  1
2,  Global Print,  1
```

- Identify the most popular item/service based on frequency.

```
Useful reports:
1. Total item interactions per customer
2. Most popular item
3. Most used manufacturer
Enter numerical selection or enter 0 to leave
2
Report: Most popular items
ProductID,  type,  color,  InteractionCount
20,  gZDbQvB9LR,  Lime,  1
19,  0k9inI0NXV,  Olive,  1
18,  J1Ibl82PbX,  Black,  1
17,  oInVwiHl2T,  coral,  1
16,  UjKHRUIBTk,  Champagne,  1
15,  2NyAsBenXY,  Brown,  1
14,  z9DZ4v7ZeD,  Olive,  1
13,  0zu0mvhNGA,  Silver,  1
12,  NRfexIjtjZ,  Cerise,  1
11,  kHckqNNzXS,  White,  1
10,  xM1EwsUrNy,  Purple,  1
9,  EawUvrHn6c,  Aquamarine,  1
8,  aPLwmXmRKz,  Azure,  1
7,  vfwXRbs2bI,  Auburn,  1
6,  g4nt0oDMjK,  Auburn,  1
5,  46o1Xr8zjK,  Fuchsia,  1
4,  VxmhCquM3S,  Brown,  1
3,  QjKN1WEjQg,  Fuchsia,  1
2,  AXwRN58jRw,  Rosewood,  1
1,  IJIOy0uZIl,  Orange,  1
```

- Find the most frequently used manufacturer.

```
1. Total item interactions per customer
2. Most popular item
3. Most used manufacturer
Enter numerical selection or enter 0 to leave
3
Report: Most popular manufacturers
Manufacturer,  InteractionCount
Crescent Paper,  1
Graphic Packaging,  2
Ohio Paper Tube,  2
Ohio Pulp Mills,  1
Roosevelt Paper,  3
International Paper,  2
Buckeye Paper,  4
Millcraft,  5
```

Avoiding SQL Injection

The most important thing to keep in mind to avoid SQL injection is to make sure prepared statements are used. An example of this is shown below.

```
String sql = "INSERT INTO Employee (employee_no, SSN, Name, Email, Tenure, Salary, Performance, pos, deptName) VALUES (?,?,?,?,?,?,?,?,?)";
ps = conn.prepareStatement(sql);

ps.setInt(1, empNum);
ps.setString(2, ssn);
ps.setString(3, name);
ps.setString(4, email);
ps.setInt(5, tenure);
ps.setInt(6, salary);
ps.setInt(7, performance);
ps.setString(8, positionName);
ps.setString(9, departmentName);
```

```
}

private static void deleteEmployee(BufferedReader input) {
    System.out.println("Enter Employee ID of employee to be deleted: ");
    int deleteID = 0;

    try {
        deleteID = Integer.parseInt(input.readLine());

        try {
            String sql = "DELETE FROM Employee WHERE employee_no= ?";
            ps = conn.prepareStatement(sql);
            ps.setInt(1, deleteID);
            int rowNum = ps.executeUpdate();

            if (rowNum > 0) {
                System.out.print("Employee successfully deleted");
            }
```