Dunder Mifflin Database Project

Final Project
CS 5318
Spring 2021

Team Members

Andrew Carpenter
Umreen Imam
Jon Snyder

Abstract

There currently is no well-structured database format that provides clear and concise insight into the world of Dunder Mifflin Paper Company in Scranton, PA. By creating an in-depth database management system, we will be capable of consolidating the office's relevant information, data, and client lists in an efficient manner at a convenient and centralized location.

This will be managed on a database hosted by AWS, utilizing various levels of SQL to accurately model the company and its numerous aspects. The desired outcome of this undertaking will provide effective means to access, manage, and modify Dunder Mifflin's internal information and data.

Mission Statement

The team has set out to develop a functional database and management system for the primary use and modeling of the Scranton, PA Dunder Mifflin Paper Company. It will be easily accessible, user-friendly, and will be utilizing data and information gathered from various sources regarding the office branch. It will model the staff, clientele, sales and purchases, human resource (HR) incidents, and the office environment.

Mission Objectives

The mission objectives for this project are as follows:

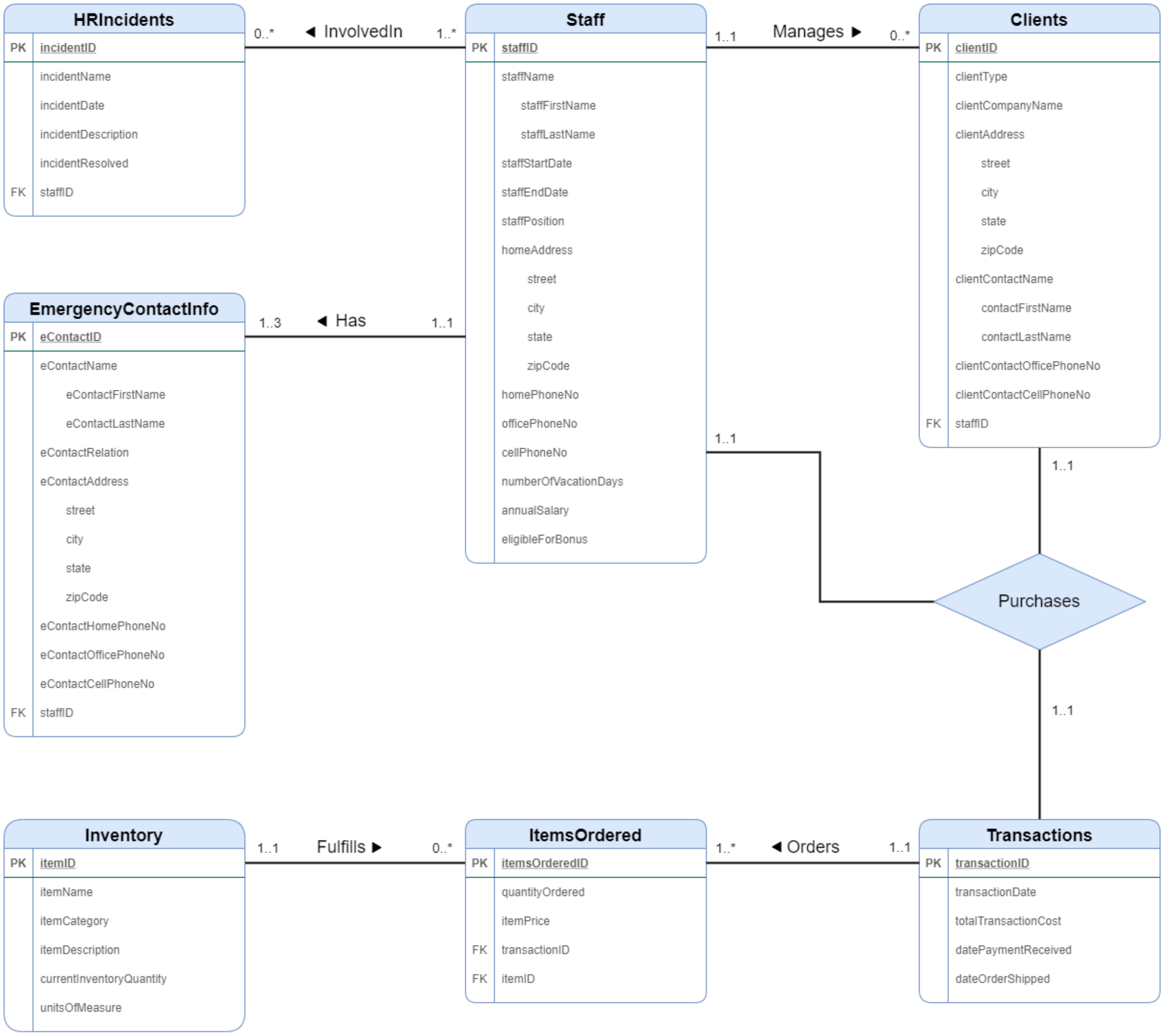
- To develop a working, functional database with logical structure.
- To have storage, modification, and accessibility functionalities.
- To have the database store information about staff, emergency contacts, HR incidents, clients, transactions, items purchased, and inventory.

- To maintain (insert, update, and delete) data on staff.
- To maintain (insert, update, and delete) data on emergency contacts.
- To maintain (insert, update, and delete) data on HR incidents.
- To maintain (insert, update, and delete) data on clients.
- To maintain (insert, update, and delete) data on transactions.
- To maintain (insert, update, and delete) data on items ordered.
- To maintain (insert, update, and delete) data on inventory.

- To report on staff.
- To report on emergency contacts.
- To report on HR incidents.
- To report on clients.
- To report on transactions.
- To report on items ordered.
- To report on inventory.

ER Diagram

The diagram below is the entity-relation diagram used to create the database for this project. The primary key and foreign key attributes are identified with PK or FK on the left-hand side of each table.



Relational Model

After converting the ER diagram to a relational model and normalizing to 3NF, the database has the following tables.

- SalaryStaff
- HourlyStaff
- HRIncidents
- InvolvedIn
- EmergencyContactInfo
- Clients
- Transactions
- Purchases
- Inventory
- ItemsOrdered

The following section contains the relational model for each table after normalization. For each table the attribute dependencies are also displayed.

Table: Staff

Relational Model

Staff (<u>staffID</u>, staffFirstName, staffLastName, staffStartDate, staffEndDate, staffPosition, street, city, state, zipCode, homePhoneNo, officePhoneNo, cellPhoneNo, numberOfVacationDays, annualSalary, eligibleForBonus)
Primary Key staffID

Attribute Dependencies

staffID → staffFirstName, staffLastName, staffStartDate, staffEndDate, staffPosition, street, city, state, zipCode, homePhoneNo, officePhoneNo, cellPhoneNo, numberOfVacationDays, annualSalary, eligibleForBonus

This table is in 3NF because it does not contain any transitive dependencies, it does not contain any partial dependencies, and each cell only contains only one value.

Table: HRIncidents

Relational Model

HRIncidents (<u>incidentID</u>, incidentName, incidentDate, incidentDescription, incidentResolved)

Primary Key incidentID

Attribute Dependencies

incidentID → incidentName, incidentDate, incidentDescription, incidentResolved

This table is in 3NF because it does not contain any transitive dependencies, it does not contain any partial dependencies, and each cell only contains only one value.

Table: InvolvedIn

Relational Model

InvolvedIn (incidentID, staffID)
Primary Key incidentID, staffID
Foreign Key incidentID **references** HRIncidents(incidentID)
Foreign Key staffID **references** Staff(staffID)

Attribute Dependencies

incidentID, staffID → none

This table is in 3NF because it does not contain any transitive dependencies, it does not contain any partial dependencies, and each cell only contains only one value.

Table: EmergencyContactInfo

Relational Model

EmergencyContactInfo (eContactID, eContactFirstName, eContactLastName, eContactRelation, street, city, state, zipCode, eContactHomePhoneNo, eContactOfficePhoneNo, eContactCellPhoneNo, staffID)
Primary Key eContactID
Foreign Key staffID **references** Staff(staffID)

Attribute Dependencies

eContactID → eContactFirstName, eContactLastName, eContactRelation, street, city, state, zipCode, eContactHomePhoneNo, eContactOfficePhoneNo, eContactCellPhoneNo, staffID

This table is in 3NF because it does not contain any transitive dependencies, it does not contain any partial dependencies, and each cell only contains only one value.

Table: Clients

Relational Model

Clients (<u>clientID</u>, clientType, clientCompanyName, street, city, state, zipCode, contactFirstName, contactLastName, clientContactOfficePhoneNo, clientContactCellPhoneNo, staffID)
Primary Key clientID
Alternate Key clientCompanyName
Foreign Key staffID **references** Staff(staffID)

Attribute Dependencies

clientID → clientType, clientCompanyName, street, city, state, zipCode, contactFirstName, contactLastName, clientContactOfficePhoneNo, clientContactCellPhoneNo, staffID

This table is in 3NF because it does not contain any transitive dependencies, it does not contain any partial dependencies, and each cell only contains only one value.

Table: Transactions

Relational Model

Transactions (<u>transactionID</u>, transactionDate, totalTransactionCost, datePaymentReceived, dateOrderShipped)
Primary Key transactionID

Attribute Dependencies

transactionID → transactionDate, totalTransactionCost, datePaymentReceived, dateOrderShipped

This table is in 3NF because it does not contain any transitive dependencies, it does not contain any partial dependencies, and each cell only contains only one value.

Table: Purchases

Relational Model

Purchases (transactionID, clientID, staffID)

Primary Key transactionID, clientID, staffID

Foreign Key transactionID **references** Transaction(transactionID)

Foreign Key clientID **references** Clients(clientID)

Foreign Key staffID **references** Staff(staffID)

Attribute Dependencies

transactionID, clientID, staffID → none

This table is in 3NF because it does not contain any transitive dependencies, it does not contain any partial dependencies, and each cell only contains only one value.

Table: Inventory

Relational Model

Inventory (itemID, itemName, itemCategory, itemDescription, currentInventoryQuantity, unitsOfMeasure)

Primary Key itemID

Attribute Dependencies

itemID → itemName, itemCategory, itemDescription, currentInventoryQuantity, unitsOfMeasure

This table is in 3NF because it does not contain any transitive dependencies, it does not contain any partial dependencies, and each cell only contains only one value.

Table: ItemsOrdered

Relational Model

ItemsOrdered (itemsOrderedID, quantityOrdered, itemPrice, transactionID, itemID)

Primary Key itemsOrderedID

Foreign Key transactionID **references** Transaction(transactionID)

Foreign Key itemID **references** Inventory(itemID)

Attribute Dependencies

itemsOrderedID → quantityOrdered, itemPrice, transactionID, itemID

This table is in 3NF because it does not contain any transitive dependencies, it does not contain any partial dependencies, and each cell only contains only one value.

Use Cases, Realizations, and Test Results

The following section contains a complete list of the use cases, step-by-step directions, SQL realizations, test expectations, and test results for each use case.

Use Case: Add new staff

Actor: HR staff or manager
Steps:

1. Actor clicks the button labeled “New staff”;
2. A new staff ID number is generated and displayed;
3. The actor enters the staff name, start date, title, address, phone numbers, and vacation information;
4. Actor then enters details for salary and if they are eligible for a bonus;
5. All information is displayed back to the user;
6. A message is displayed asking for the user to confirm the new staff details;
7. User clicks the button labeled “Confirm”

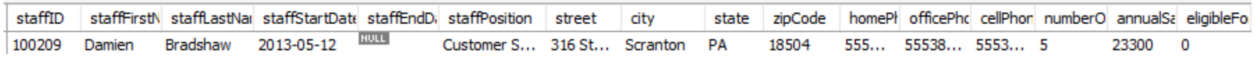
Use Case Realization:

```
INSERT INTO dunderDB.Staff (staffID, staffFirstName, staffLastName,
     staffStartDate, staffEndDate, staffPosition, street, city,
     state, zipCode, homePhoneNo, officePhoneNo, cellPhoneNo,
     numberOfVacationDays, annualSalary, eligibleForBonus)
VALUES (100209, 'Damien', 'Bradshaw', '2013-05-12', NULL,
     'Customer Service', '316 Steven Rd', 'Scranton', 'PA',
     18504, 5553828491, 5553828423, 5553826483, 5, 23300, 0);
SELECT * FROM dunderDB.Staff WHERE staffID = 100209;
```

Test Plan and Records:
INPUT: This use case requires the following inputs: {staffID, staffFirstName, staffLastName, staffStartDate, staffEndDate, staffPosition, street, city, state, zipCode, homePhoneNo, officePhoneNo, cellPhoneNo, numberOfVacationDays, annualSalary, eligibleForBonus}

EXPECTED: We expect to see a new staff member added to the table and displayed to the user

ACTUAL:



staffID	staffFirstN	staffLastNa	staffStartDate	staffEndDa	staffPosition	street	city	state	zipCode	homePl	officePhc	cellPhor	numberO	annualSa	eligibleFo
100209	Damien	Bradshaw	2013-05-12	NULL	Customer S...	316 St...	Scranton	PA	18504	555...	55538...	5553...	5	23300	0

Use Case: Update staff information

Actor: HR staff or manager
Steps:

1. Actor clicks the button labeled “Update staff information”;
2. Prompt asks for staff ID number;
3. Actor enters staff ID number;
4. All information for the referenced staff is displayed;
5. Actor edits the information that needs to be updated;
6. The updated information is displayed back to the user;
7. A message is displayed asking for the user to confirm the staff details;
8. User clicks the button labeled “Confirm”

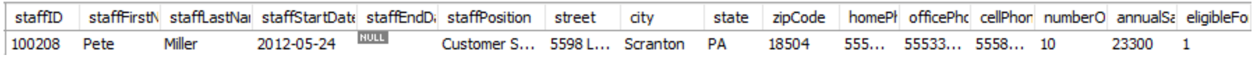
Use Case Realization:

```
UPDATE dunderDB.Staff
SET numberOfVacationDays = 10, eligibleForBonus = 1
WHERE staffID = 100208;
SELECT * FROM dunderDB.Staff WHERE staffID = 100208;
```

Test Plan and Records:
INPUT: This use case requires the following input: staff ID number and any additional columns that will be updated depending on the specific case

EXPECTED: We expect to see the specified staff member’s information with updated information displayed to the user

ACTUAL:



staffID	staffFirstN	staffLastNa	staffStartDate	staffEndDa	staffPosition	street	city	state	zipCode	homePh	officePhc	cellPhon	numberO	annualSa	eligibleFo
100208	Pete	Miller	2012-05-24	NULL	Customer S...	5598 L...	Scranton	PA	18504	555...	55533...	5558...	10	23300	1

Use Case: Count the number of current active staff

Actor: HR staff or manager
Steps:

1. Actor clicks the button labeled “Count current staff”;
2. Database calculates the total number of staff that do not have a value in the attribute “staffEndDate”;
3. The result is displayed to the user

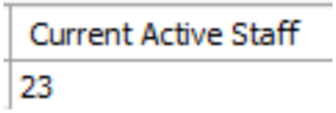
Use Case Realization:

```
SELECT COUNT(staffID) AS 'Current Active Staff'
FROM dunderDB.Staff
WHERE staffEndDate IS NULL;
```

Test Plan and Records:
INPUT: None, selection only

EXPECTED: We expect to see the current count of employees displayed to the user

ACTUAL:



Current Active Staff
23

Use Case: Display all sales for each staff member

Actor: Management, sales, accounting, or warehouse staff
Steps:

1. Actor clicks the button labeled “Display all sales for all staff members”;
2. Database joins the transaction details with their related items ordered details;
3. The result is displayed to the user

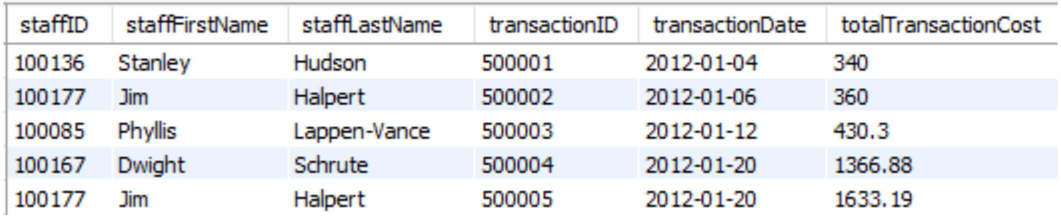
Use Case Realization:

```
SELECT s.staffID, s.staffFirstName, s.staffLastName,
     t.transactionID, t.transactionDate, t.totalTransactionCost
FROM dunderDB.Staff s, dunderDB.Transactions t,
     dunderDB.Purchases p
WHERE s.staffID = p.staffID AND p.transactionID = t.transactionID;
```

Test Plan and Records:
INPUT: None, Selection only

EXPECTED: We expect a table that shows all staff who were involved in a transaction. The table should show the staff ID, staff first name, staff last name, transaction ID, transaction date, and transaction total cost.

ACTUAL:



staffID	staffFirstName	staffLastName	transactionID	transactionDate	totalTransactionCost
100136	Stanley	Hudson	500001	2012-01-04	340
100177	Jim	Halpert	500002	2012-01-06	360
100085	Phyllis	Lappen-Vance	500003	2012-01-12	430.3
100167	Dwight	Schrute	500004	2012-01-20	1366.88
100177	Jim	Halpert	500005	2012-01-20	1633.19

Use Case: Delete a member of staff

Actor: HR staff or manager
Steps:

1. Actor clicks the button labeled “Permanently delete a member of staff”;
2. Prompt asks for staff ID number;
3. Actor enters staff ID number;
4. All information for the referenced staff member is displayed;
5. A message is displayed asking for the user to confirm that this will be deleted from the database;
6. User clicks the button labeled “Confirm”

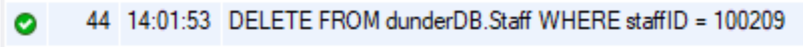
Use Case Realization:

```
DELETE FROM dunderDB.Staff
WHERE staffID = 100209;
```

Test Plan and Records:
INPUT: This use case requires the following input: staff ID number

EXPECTED: We expect a member of staff and all of their related information deleted from the database

ACTUAL:



```
✓  44  14:01:53  DELETE FROM dunderDB.Staff WHERE staffID = 100209
```

Use Case: Add emergency contact

Actor: HR staff or manager
Steps:

1. Actor clicks the button labeled “New emergency contact”;
2. A new emergency contact ID number is generated and displayed;
3. The actor enters the name, relationship, address, phone numbers, and staff ID number for the new emergency contact;
4. All information is displayed back to the user;
5. A message is displayed asking for the user to confirm the new emergency contact details;
6. User clicks the button labeled “Confirm”

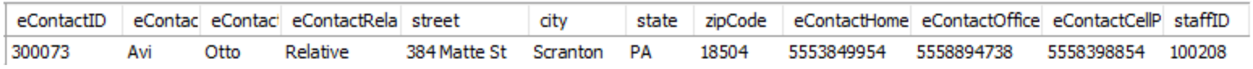
Use Case Realization:

```
INSERT INTO dunderDB.EmergencyContactInfo (eContactID,
     eContactFirstName, eContactLastName,
     eContactRelation, street, city,
     state, zipCode, eContactHomePhoneNo,
     eContactOfficePhoneNo, eContactCellPhoneNo, staffID)
VALUES(300073, 'Avi', 'Otto', 'Relative', '384 Matte St',
     'Scranton', 'PA', 18504, 5553849954, 5558894738, 5558398854,
     100208);
SELECT * FROM dunderDB.EmergencyContactInfo WHERE eContactID =
     300073;
```

Test Plan and Records:
INPUT: This use case requires the following inputs: {eContactID, eContactFirstName, eContactLastName, eContactRelation, street, city, state, zipCode, eContactHomePhoneNo, eContactOfficePhoneNo, eContactCellPhoneNo, staffID}

EXPECTED: We expect to have a new emergency contact added to the EmergencyContactInfo table and displayed to the user

ACTUAL:



eContactID	eContac	eContac	eContactRela	street	city	state	zipCode	eContactHome	eContactOffice	eContactCellP	staffID
300073	Avi	Otto	Relative	384 Matte St	Scranton	PA	18504	5553849954	5558894738	5558398854	100208

Use Case: Update emergency contact details

Actor: HR staff or manager
Steps:

1. Actor clicks the button labeled “Update emergency contact details”;
2. Prompt asks for emergency contact ID number;
3. Actor enters emergency contact ID number;
4. All information for the referenced emergency contact is displayed;
5. Actor edits the information that needs to be updated;
6. The updated information is displayed back to the user;
7. A message is displayed asking for the user to confirm the staff details;
8. User clicks the button labeled “Confirm”

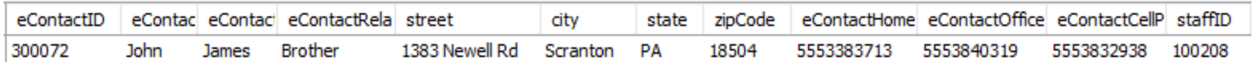
Use Case Realization:

```
UPDATE dunderDB.EmergencyContactInfo
SET eContactFirstName = 'John', eContactLastName = 'James',
     eContactRelation = 'Brother', street = '1383 Newell Rd',
     eContactHomePhoneNo = 5553383713, eContactOfficePhoneNo =
     5553840319, eContactCellPhoneNo = 5553832938
WHERE eContactID = 300072;
SELECT * FROM dunderDB.EmergencyContactInfo WHERE eContactID =
     300072;
```

Test Plan and Records:
INPUT: This use case requires the following input: emergency contact ID number and update information

EXPECTED: We expect the specified emergency contact’s information to be updated with the new information and displayed to the user

ACTUAL:



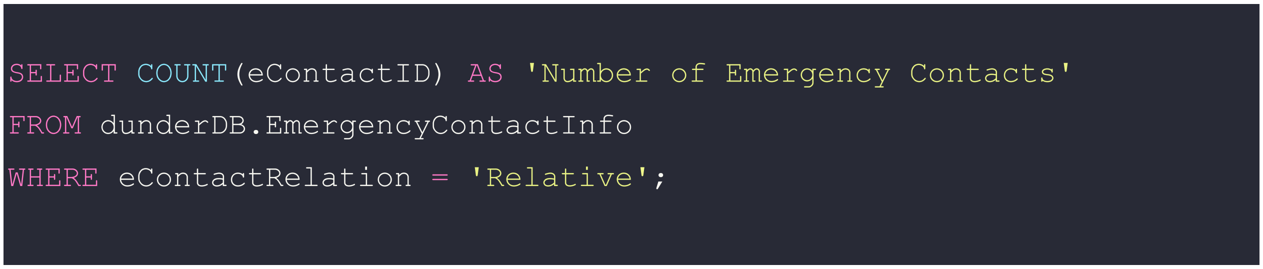
eContactID	eContac	eContac	eContactRela	street	city	state	zipCode	eContactHome	eContactOffice	eContactCellP	staffID
300072	John	James	Brother	1383 Newell Rd	Scranton	PA	18504	5553383713	5553840319	5553832938	100208

Use Case: Count number of emergency contacts by relation type

Actor: HR staff or manager
Steps:

1. Actor clicks the button labeled “Count emergency contacts by relationship type”;
2. Prompt asks for the relationship type of interest for the query;
3. Actor enters the relationship type that they want used for the query;
4. Database calculates the number of emergency contacts that have the specified relationship type;
5. The result is displayed to the user

Use Case Realization:

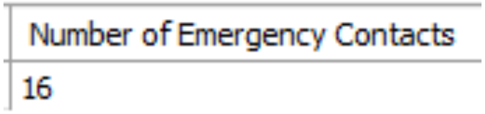


```
SELECT COUNT(eContactID) AS 'Number of Emergency Contacts'
FROM dunderDB.EmergencyContactInfo
WHERE eContactRelation = 'Relative';
```

Test Plan and Records:
INPUT: This use case requires the following input: emergency contact relationship type

EXPECTED: We expect to see a current count of emergency contacts with a specific type of relationship displayed to the user

ACTUAL:



Number of Emergency Contacts
16

Use Case: Display all emergency contacts for all staff members

Actor: HR staff and management
Steps:

1. Actor clicks the button labeled “Display all emergency contacts for all staff member”;
2. Database joins the emergency contact details with their related staff details;
3. The result is displayed to the user

Use Case Realization:

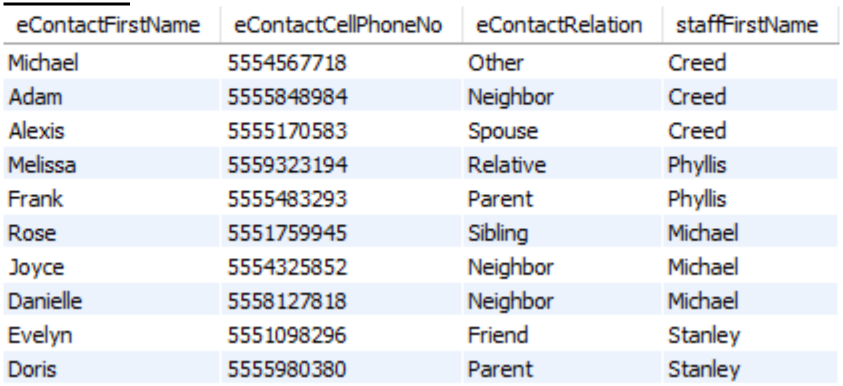
```
SELECT EmergencyContactInfo.eContactFirstName,
     EmergencyContactInfo.eContactCellPhoneNo,
     EmergencyContactInfo.eContactRelation, Staff.staffFirstName
FROM dunderDB.EmergencyContactInfo
LEFT JOIN Staff
ON EmergencyContactInfo.staffID = Staff.staffID
LIMIT 10;
```

Note: This code is limited to show 10 results for the purposes of this report.

Test Plan and Records:
INPUT: None, Selection only

EXPECTED: We expect a table with all emergency contacts and their information displayed to the user

ACTUAL:



eContactFirstName	eContactCellPhoneNo	eContactRelation	staffFirstName
Michael	5554567718	Other	Creed
Adam	5555848984	Neighbor	Creed
Alexis	5555170583	Spouse	Creed
Melissa	5559323194	Relative	Phyllis
Frank	5555483293	Parent	Phyllis
Rose	5551759945	Sibling	Michael
Joyce	5554325852	Neighbor	Michael
Danielle	5558127818	Neighbor	Michael
Evelyn	5551098296	Friend	Stanley
Doris	5555980380	Parent	Stanley

Use Case: Delete an emergency contact

Actor: HR staff or manager
Steps:

1. Actor clicks the button labeled “Permanently delete an emergency contact”;
2. Prompt asks for emergency contact ID number;
3. Actor enters emergency contact ID number;
4. All information for the referenced emergency contact is displayed;
5. A message is displayed asking for the user to confirm that this will be deleted from the database;
6. User clicks the button labeled “Confirm”

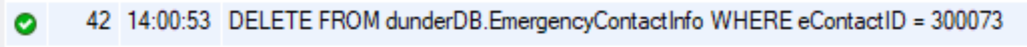
Use Case Realization:

```
DELETE FROM dunderDB.EmergencyContactInfo
WHERE eContactID = 300073;
```

Test Plan and Records:
INPUT: This use case requires the following input: emergency contact ID number

EXPECTED: We expect to delete all emergency contact related information from the database for the specified emergency contact

ACTUAL:



```
✓  42  14:00:53  DELETE FROM dunderDB.EmergencyContactInfo WHERE eContactID = 300073
```

Use Case: Add a new HR incident

Actor: HR staff
Steps:

1. Actor clicks the button labeled “New HR incident”;
2. A new HR incident ID number is generated and displayed;
3. The actor enters the incident name, incident date, description of the incident, and if the incident has been resolved or not;
4. Prompt asks for staff ID numbers or staff names for anyone involved in the incident;
5. Actor enters staff ID numbers or staff names for anyone involved in the incident;
6. All information is displayed back to the user;
7. A message is displayed asking for the user to confirm the HR incident details;
8. User clicks the button labeled “Confirm”

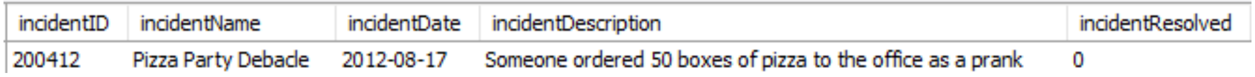
Use Case Realization:

```
INSERT INTO dunderDB.HRIncidents (incidentID, incidentName,
     incidentDate, incidentDescription, incidentResolved)
VALUES(200412, 'Pizza Party Debacle', '2012-08-17',
     'Someone ordered 50 boxes of pizza to the office as a prank',
     0);
INSERT INTO dunderDB.InvolvedIn (incidentID, staffID)
VALUES(200412, 100182);
SELECT * FROM dunderDB.HRIncidents WHERE incidentID = 200412;
```

Test Plan and Records:
INPUT: This use case requires the following inputs: {incidentID, incidentName, incidentDate, incidentDescription, incidentResolved, staffID}

EXPECTED: We expect a new HR Incident to be added to the HR Incident table and displayed to the user

ACTUAL:



incidentID	incidentName	incidentDate	incidentDescription	incidentResolved
200412	Pizza Party Debacle	2012-08-17	Someone ordered 50 boxes of pizza to the office as a prank	0

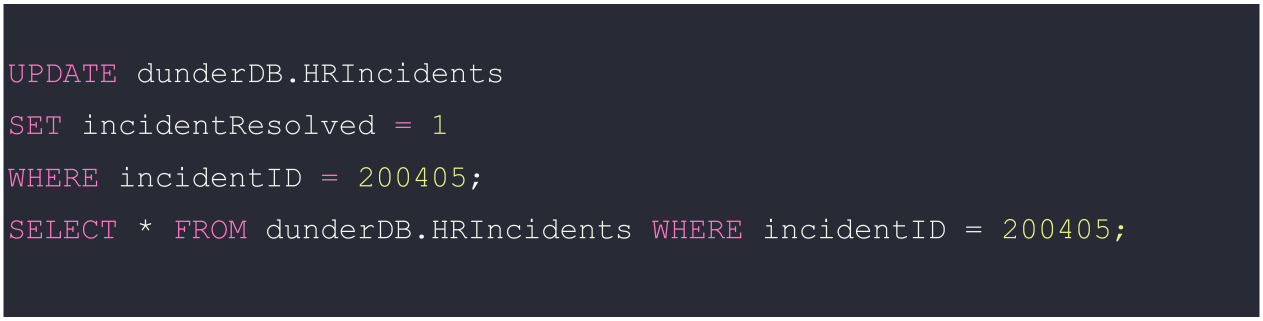
Use Case: Update details on an HR incident

Actor: HR staff

Steps:

1. Actor clicks the button labeled “Update HR incident”;
2. Prompt asks for HR incident ID number;
3. Actor enters HR incident ID number;
4. All information for the referenced HR incident is displayed;
5. Actor edits the information that needs to be updated;
6. The updated information is displayed back to the user;
7. A message is displayed asking for the user to confirm the HR incident details;
8. User clicks the button labeled “Confirm”

Use Case Realization:



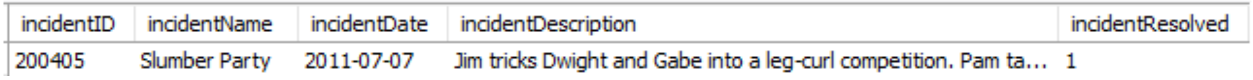
```
UPDATE dunderDB.HRIncidents
SET incidentResolved = 1
WHERE incidentID = 200405;
SELECT * FROM dunderDB.HRIncidents WHERE incidentID = 200405;
```

Test Plan and Records:

INPUT: This use case requires the following input: HR Incident ID number and the pertinent information to update about this specific HR incident

EXPECTED: We expect to see the specified HR incident’s information updated and displayed to the user

ACTUAL:



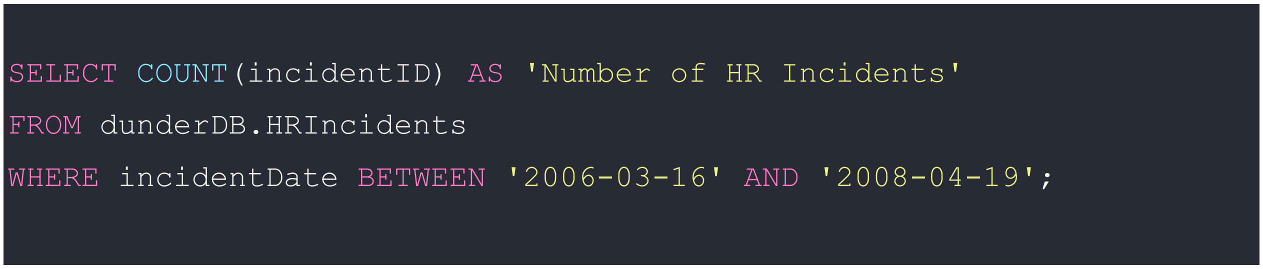
incidentID	incidentName	incidentDate	incidentDescription	incidentResolved
200405	Slumber Party	2011-07-07	Jim tricks Dwight and Gabe into a leg-curl competition. Pam ta...	1

Use Case: Count number of HR incidents between two dates

Actor: HR staff
Steps:

1. Actor clicks the button labeled “Number of HR incidents between two dates”;
2. Prompt asks for the start date and end date to be used for the query;
3. Actor enters dates that they want used for the query;
4. Database calculates the number of HR incidents that occurred between the two dates;
5. The result is displayed to the user

Use Case Realization:



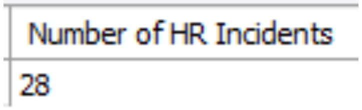
```
SELECT COUNT(incidentID) AS 'Number of HR Incidents'
FROM dunderDB.HRIncidents
WHERE incidentDate BETWEEN '2006-03-16' AND '2008-04-19';
```

Test Plan and Records:

INPUT: This use case requires the following inputs: start date and end date

EXPECTED: We expect an accurate count of number of HR incidents between the specified dates displayed to the user

ACTUAL:



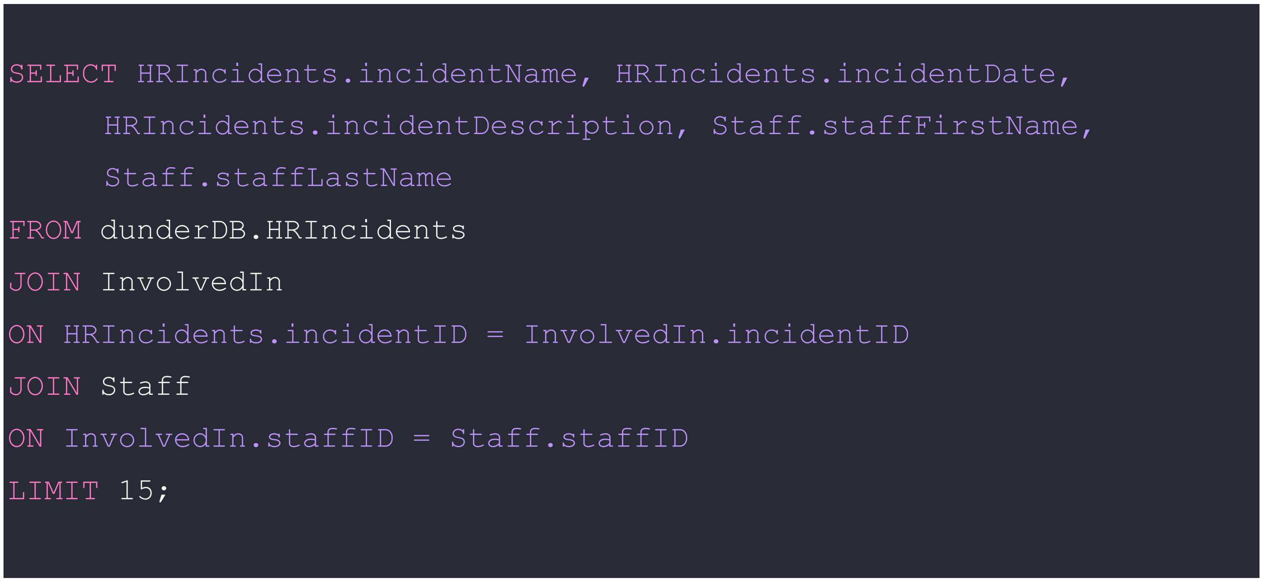
Number of HR Incidents
28

Use Case: Display all HR incidents for all staff members

Actor: HR staff
Steps:

1. Actor clicks the button labeled “Display all HR incidents for all staff member”;
2. Database joins the HR incident details with their related staff details;
3. The result is displayed to the user

Use Case Realization:



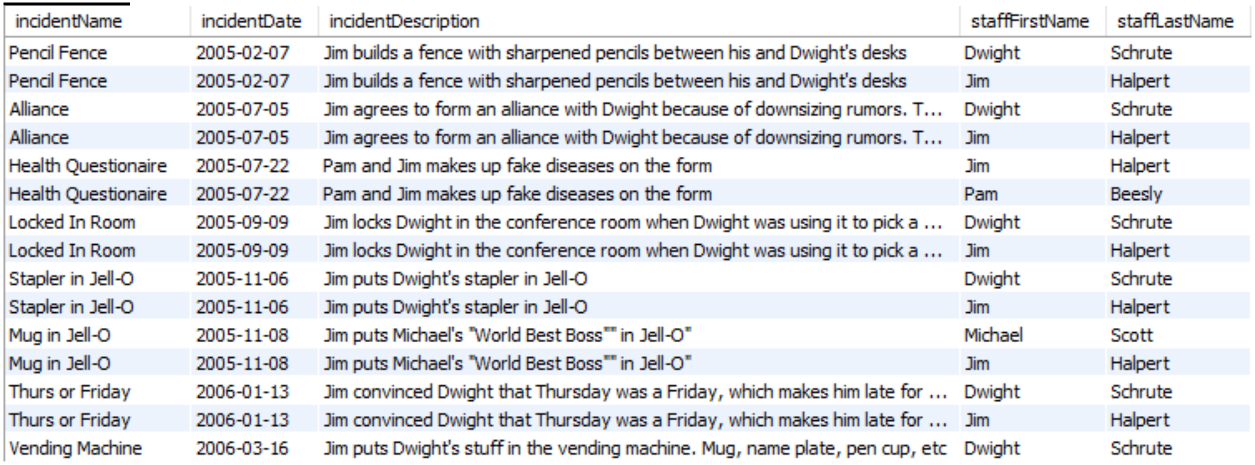
```
SELECT HRIncidents.incidentName, HRIncidents.incidentDate,
     HRIncidents.incidentDescription, Staff.staffFirstName,
     Staff.staffLastName
FROM dunderDB.HRIncidents
JOIN InvolvedIn
ON HRIncidents.incidentID = InvolvedIn.incidentID
JOIN Staff
ON InvolvedIn.staffID = Staff.staffID
LIMIT 15;
```

Note: This code is limited to show 15 results for the purposes of this report.

Test Plan and Records:
INPUT: None, Selection only

EXPECTED: We expect a table with all HR incidents and related staff displayed to the user

ACTUAL:



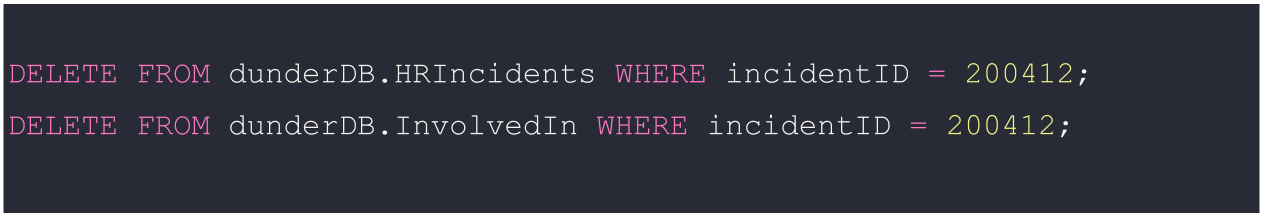
incidentName	incidentDate	incidentDescription	staffFirstName	staffLastName
Pencil Fence	2005-02-07	Jim builds a fence with sharpened pencils between his and Dwight's desks	Dwight	Schrute
Pencil Fence	2005-02-07	Jim builds a fence with sharpened pencils between his and Dwight's desks	Jim	Halpert
Alliance	2005-07-05	Jim agrees to form an alliance with Dwight because of downsizing rumors. T...	Dwight	Schrute
Alliance	2005-07-05	Jim agrees to form an alliance with Dwight because of downsizing rumors. T...	Jim	Halpert
Health Questionaire	2005-07-22	Pam and Jim makes up fake diseases on the form	Jim	Halpert
Health Questionaire	2005-07-22	Pam and Jim makes up fake diseases on the form	Pam	Beesly
Locked In Room	2005-09-09	Jim locks Dwight in the conference room when Dwight was using it to pick a ...	Dwight	Schrute
Locked In Room	2005-09-09	Jim locks Dwight in the conference room when Dwight was using it to pick a ...	Jim	Halpert
Stapler in Jell-O	2005-11-06	Jim puts Dwight's stapler in Jell-O	Dwight	Schrute
Stapler in Jell-O	2005-11-06	Jim puts Dwight's stapler in Jell-O	Jim	Halpert
Mug in Jell-O	2005-11-08	Jim puts Michael's "World Best Boss"" in Jell-O"	Michael	Scott
Mug in Jell-O	2005-11-08	Jim puts Michael's "World Best Boss"" in Jell-O"	Jim	Halpert
Thurs or Friday	2006-01-13	Jim convinced Dwight that Thursday was a Friday, which makes him late for ...	Dwight	Schrute
Thurs or Friday	2006-01-13	Jim convinced Dwight that Thursday was a Friday, which makes him late for ...	Jim	Halpert
Vending Machine	2006-03-16	Jim puts Dwight's stuff in the vending machine. Mug, name plate, pen cup, etc	Dwight	Schrute

Use Case: Delete an HR incident

Actor: HR staff
Steps:

1. Actor clicks the button labeled “Permanently delete an HR incident”;
2. Prompt asks for HR incident ID number;
3. Actor enters HR incident ID number;
4. All information for the referenced HR incident is displayed;
5. A message is displayed asking for the user to confirm that this will be deleted from the database;
6. User clicks the button labeled “Confirm”

Use Case Realization:



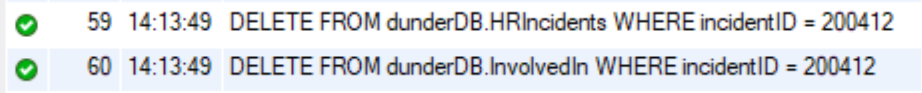
```
DELETE FROM dunderDB.HRIncidents WHERE incidentID = 200412;
DELETE FROM dunderDB.InvolvedIn WHERE incidentID = 200412;
```

Test Plan and Records:

INPUT: This use case requires the following input: HR Incident number

EXPECTED: We expect the specified HR incident and all related information deleted from the database

ACTUAL:



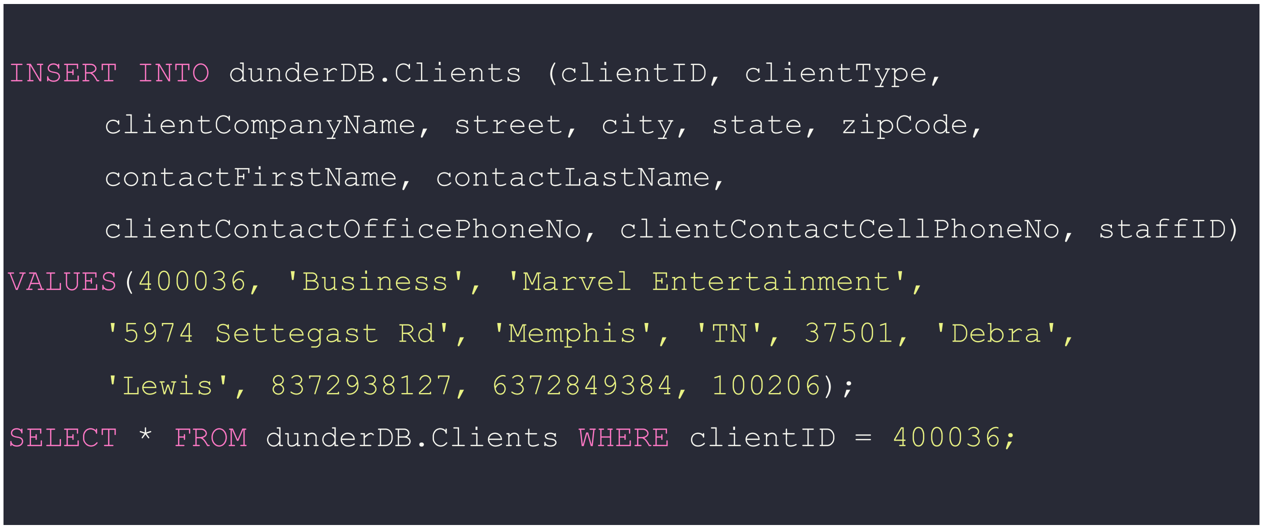
```
59  14:13:49  DELETE FROM dunderDB.HRIncidents WHERE incidentID = 200412
60  14:13:49  DELETE FROM dunderDB.InvolvedIn WHERE incidentID = 200412
```

Use Case: Add new client

Actor: Sales staff or manager
Steps:

1. Actor clicks the button labeled "Add new client";
2. A new client ID number is generated and displayed;
3. The actor enters the client name, client type, address, and office phone number;
4. Optionally, the actor can also enter a contact name, contact title, contact office phone number, and contact cell phone number;
5. All information is displayed back to the user;
6. A message is displayed asking for the user to confirm the client details;
7. User clicks the button labeled "Confirm"

Use Case Realization:

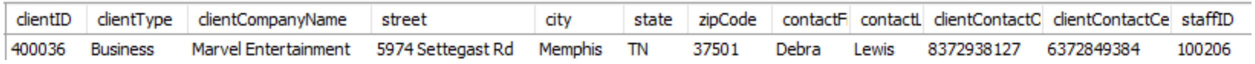


```
INSERT INTO dunderDB.Clients (clientID, clientType,
     clientCompanyName, street, city, state, zipCode,
     contactFirstName, contactLastName,
     clientContactOfficePhoneNo, clientContactCellPhoneNo, staffID)
VALUES(400036, 'Business', 'Marvel Entertainment',
     '5974 Settegast Rd', 'Memphis', 'TN', 37501, 'Debra',
     'Lewis', 8372938127, 6372849384, 100206);
SELECT * FROM dunderDB.Clients WHERE clientID = 400036;
```

Test Plan and Records:
INPUT: This use case requires the following inputs:{clientID, clientType, clientCompanyName, street, city, state, zipCode, contactFirstName, contactLastName, clientContactOfficePhoneNo, clientContactCellPhoneNo, staffID}

EXPECTED: We expect to see a new client to be added to the Clients table and displayed to the user

ACTUAL:



clientID	clientType	clientCompanyName	street	city	state	zipCode	contactF	contactL	clientContactC	clientContactCe	staffID
400036	Business	Marvel Entertainment	5974 Settegast Rd	Memphis	TN	37501	Debra	Lewis	8372938127	6372849384	100206

Use Case: Update client information

Actor: Sales staff or manager
Steps:

1. Actor clicks the button labeled “Update client information”;
2. Prompt asks for client ID number;
3. Actor enters client ID number;
4. All information for the referenced client is displayed;
5. Actor edits the information that needs to be updated;
6. The updated information is displayed back to the user;
7. A message is displayed asking for the user to confirm the client details;
8. User clicks the button labeled “Confirm”

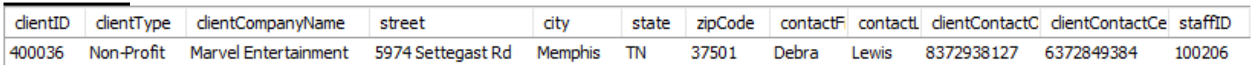
Use Case Realization:

```
UPDATE dunderDB.Clients
SET clientType = 'Non-Profit'
WHERE clientID = 400036;
SELECT * FROM dunderDB.Clients WHERE clientID = 400036;
```

Test Plan and Records:
INPUT: This use case requires the following input: client ID number and the pertinent information to update about this client

EXPECTED: We expect to see the specified client’s information updated and displayed to the user

ACTUAL:



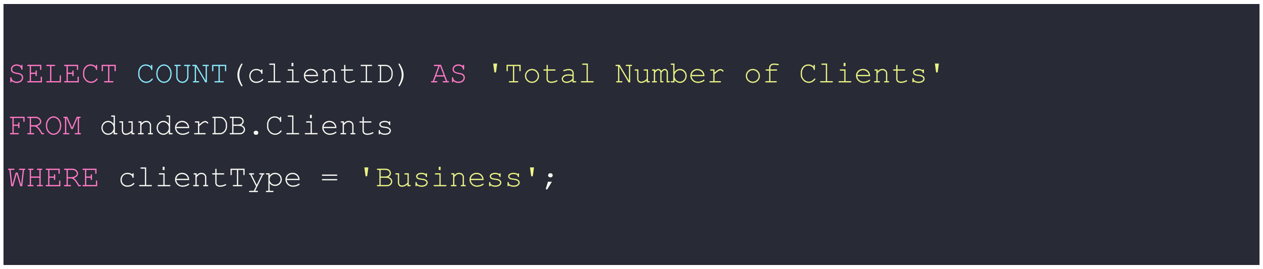
clientID	clientType	clientCompanyName	street	city	state	zipCode	contactF	contactL	clientContactC	clientContactCe	staffID
400036	Non-Profit	Marvel Entertainment	5974 Settegast Rd	Memphis	TN	37501	Debra	Lewis	8372938127	6372849384	100206

Use Case: Count number of clients of a specific client type

Actor: Sales staff or manager
Steps:

1. Actor clicks the button labeled “Number of clients by client type”;
2. Prompt asks for the client type of interest for the query;
3. Actor enters the client type that they want used for the query;
4. Database calculates the number of clients that have the specified client type;
5. The result is displayed to the user

Use Case Realization:

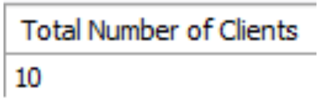


```
SELECT COUNT(clientID) AS 'Total Number of Clients'
FROM dunderDB.Clients
WHERE clientType = 'Business';
```

Test Plan and Records:
INPUT: This use case requires the following input: client type

EXPECTED: We expect to see the current count of clients of the specified type displayed to the user

ACTUAL:



Total Number of Clients
10

Use Case: Display the total number of transactions for a client

Actor: Sales staff or manager
Steps:

1. Actor clicks the button labeled “Display total purchases by client”;
2. Prompt asks for the client ID of interest for the query;
3. Actor enters the client ID that they want used for the query;
4. Database calculates the total number of transactions for the specified client ID;
5. The result is displayed to the user

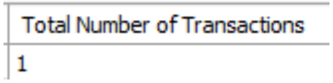
Use Case Realization:

```sql
SELECT COUNT(transactionID) AS 'Total Number of Transactions'
FROM dunderDB.Purchases
WHERE clientID = 400025;
```

Test Plan and Records:
INPUT: This use case requires the following inputs: client ID

EXPECTED: We expect an accurate count of number of number of transactions from the specified client displayed to the user

ACTUAL:



Total Number of Transactions
1

Use Case: Display all clients and their assigned sales staff

Actor: Sales staff and management
Steps:

1. Actor clicks the button labeled “Display clients and their sales staff”;
2. Database joins the client details with their related staff details;
3. The result is displayed to the user

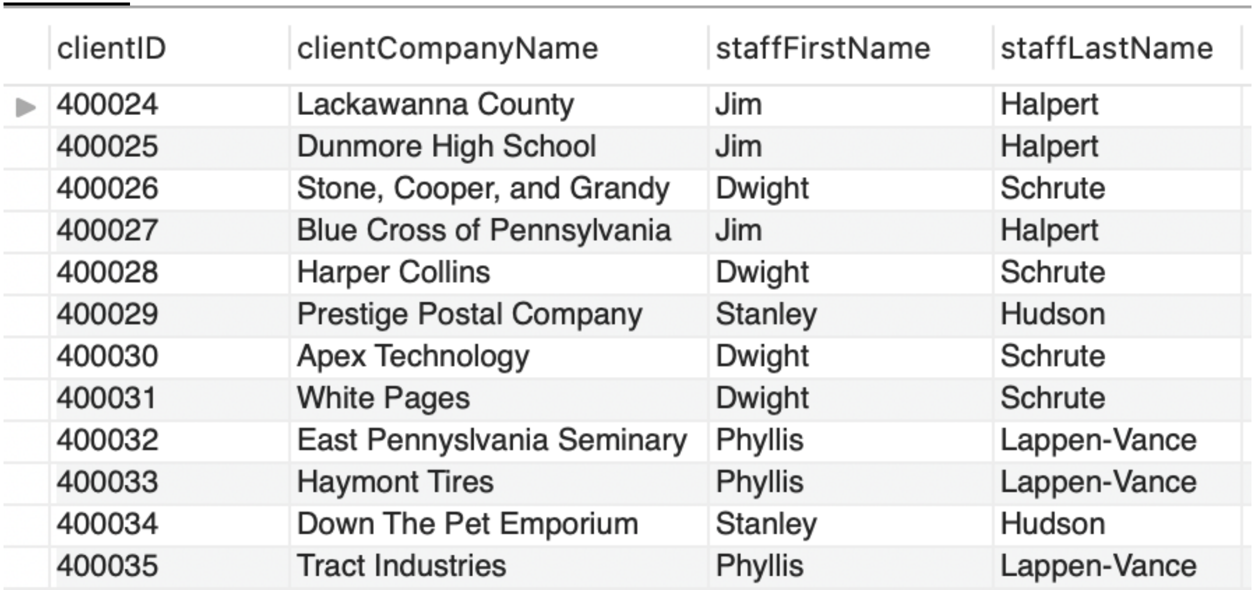
Use Case Realization:

```
SELECT Clients.clientID, Clients.clientCompanyName,
     Staff.staffFirstName, Staff.staffLastName,
FROM dunderDB.Clients
INNER JOIN Staff
ON Clients.staffID = Staff.staffID;
```

Test Plan and Records:
INPUT: None, Selection only

EXPECTED: We expect a table with all clients and the related staff displayed to the user

ACTUAL:



	clientID	clientCompanyName	staffFirstName	staffLastName
▶	400024	Lackawanna County	Jim	Halpert
	400025	Dunmore High School	Jim	Halpert
	400026	Stone, Cooper, and Grandy	Dwight	Schrute
	400027	Blue Cross of Pennsylvania	Jim	Halpert
	400028	Harper Collins	Dwight	Schrute
	400029	Prestige Postal Company	Stanley	Hudson
	400030	Apex Technology	Dwight	Schrute
	400031	White Pages	Dwight	Schrute
	400032	East Pennyslvania Seminary	Phyllis	Lappen-Vance
	400033	Haymont Tires	Phyllis	Lappen-Vance
	400034	Down The Pet Emporium	Stanley	Hudson
	400035	Tract Industries	Phyllis	Lappen-Vance

Use Case: Delete a client

Actor: Sales staff or manager
Steps:

1. Actor clicks the button labeled “Permanently delete a client”;
2. Prompt asks for client ID number;
3. Actor enters client ID number;
4. All information for the referenced client is displayed;
5. A message is displayed asking for the user to confirm that this will be deleted from the database;
6. User clicks the button labeled “Confirm”

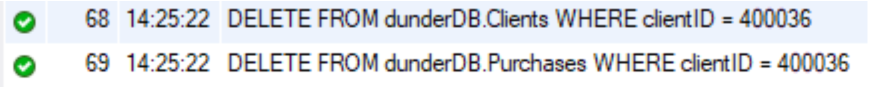
Use Case Realization:

```
DELETE FROM dunderDB.Clients WHERE clientID = 400036;
DELETE FROM dunderDB.Purchases WHERE clientID = 400036;
```

Test Plan and Records:
INPUT: This use case requires the following input: client ID number

EXPECTED: We expect the specified client and all of their related information deleted from the database

ACTUAL:



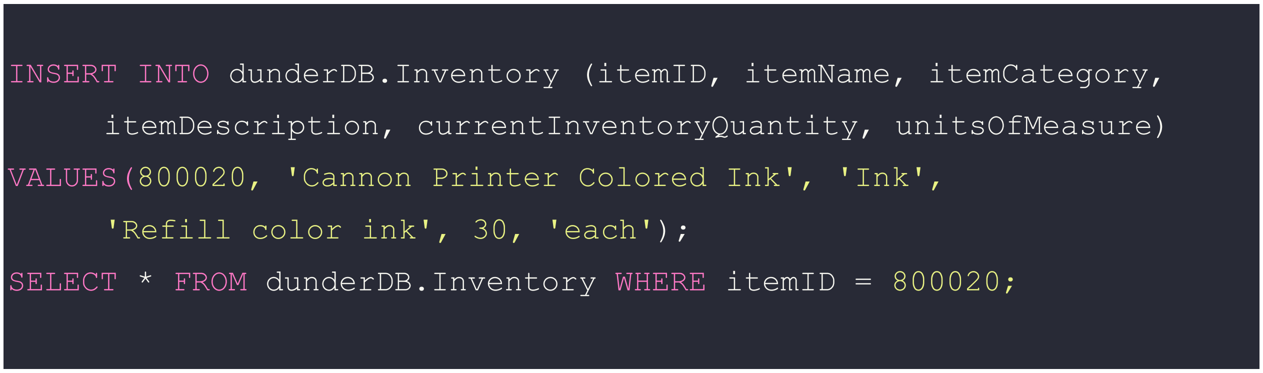
```
✓   68  14:25:22  DELETE FROM dunderDB.Clients WHERE clientID = 400036
✓   69  14:25:22  DELETE FROM dunderDB.Purchases WHERE clientID = 400036
```

Use Case: Add new item to inventory

Actor: Warehouse staff, sales staff, or manager
Steps:

1. Actor clicks the button labeled “Add new item to inventory”;
2. A new item ID number is generated and displayed;
3. The actor enters the item name, item category, item description, current inventory quantity, and units of measure;
4. All information is displayed back to the user;
5. A message is displayed asking for the user to confirm the item details;
6. User clicks the button labeled “Confirm”

Use Case Realization:

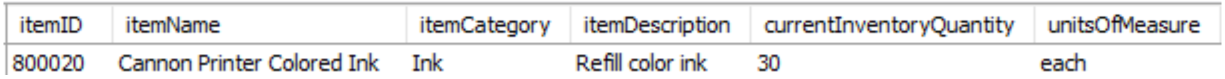


```
INSERT INTO dunderDB.Inventory (itemID, itemName, itemCategory,
     itemDescription, currentInventoryQuantity, unitsOfMeasure)
VALUES(800020, 'Cannon Printer Colored Ink', 'Ink',
     'Refill color ink', 30, 'each');
SELECT * FROM dunderDB.Inventory WHERE itemID = 800020;
```

Test Plan and Records:
INPUT: This use case requires the following inputs: { itemID, itemName, itemCategory, itemDescription, currentInventoryQuantity, unitsOfMeasure}

EXPECTED: We expect to see a new Inventory item appear in the Inventory table and displayed to the user

ACTUAL:



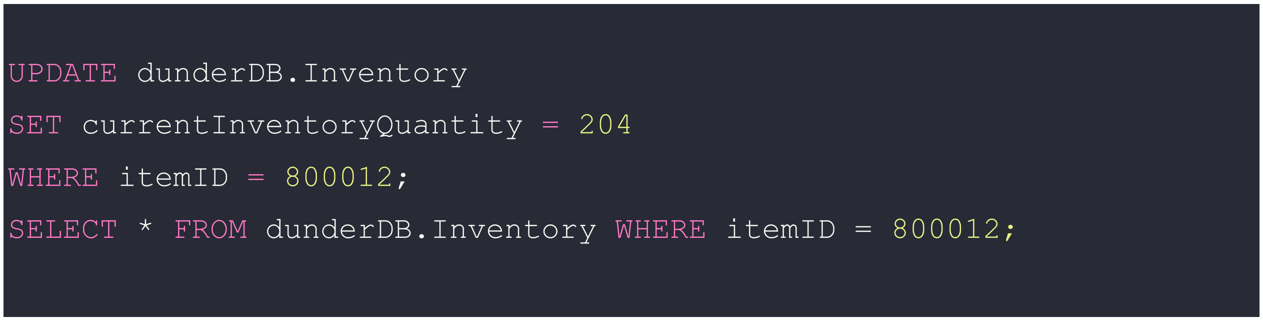
itemID	itemName	itemCategory	itemDescription	currentInventoryQuantity	unitsOfMeasure
800020	Cannon Printer Colored Ink	Ink	Refill color ink	30	each

Use Case: Update items in inventory

Actor: Warehouse staff
Steps:

1. Actor clicks the button labeled “Update inventory for an item”;
2. Prompt asks for item ID number;
3. Actor enters item ID number;
4. All information for the referenced client is displayed;
5. Actor edits the information that needs to be updated;
6. The updated information is displayed back to the user;
7. A message is displayed asking for the user to confirm the item details;
8. User clicks the button labeled “Confirm”

Use Case Realization:

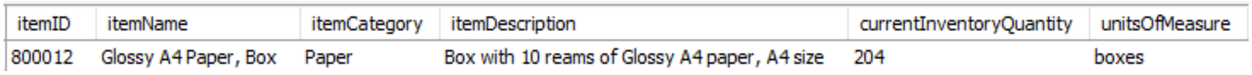


```
UPDATE dunderDB.Inventory
SET currentInventoryQuantity = 204
WHERE itemID = 800012;
SELECT * FROM dunderDB.Inventory WHERE itemID = 800012;
```

Test Plan and Records:
INPUT: This use case requires the following input: item ID and the pertinent information to update about this item in inventory

EXPECTED: We expect to see the specified item’s information updated and displayed to the user

ACTUAL:



itemID	itemName	itemCategory	itemDescription	currentInventoryQuantity	unitsOfMeasure
800012	Glossy A4 Paper, Box	Paper	Box with 10 reams of Glossy A4 paper, A4 size	204	boxes

Use Case: Count inventory quantity for a particular category

Actor: Warehouse staff, sales staff, or manager
Steps:

1. Actor clicks the button labeled “Display inventory quantity by category and units of measure”;
2. Prompt asks for the item category and units of measure of interest for the query;
3. Actor enters the item category and units of measure that they want used for the query;
4. Database calculates the quantity of items in inventory that have the specified inventory category and units of measure;
5. The result is displayed to the user

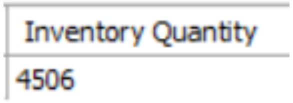
Use Case Realization:

```
SELECT SUM(currentInventoryQuantity) AS 'Inventory Quantity'
FROM dunderDB.Inventory
WHERE itemCategory = 'Paper' AND unitsOfMeasure = 'Reams';
```

Test Plan and Records:
INPUT: This use case requires the following input: item category of interest

EXPECTED: We expect a total quantity of products existing in inventory displayed to the user

ACTUAL:



Inventory Quantity
4506

Use Case: Display all transactions by an item category

Actor: Warehouse staff, sales staff, or manager
Steps:

1. Actor clicks the button labeled “Display all transactions by an item category”;
2. Prompt asks for the item category of interest for the query;
3. Actor enters the item category that they want used for the query;
4. Database calculates the quantity of items in inventory that have the specified inventory category;
5. The result is displayed to the user

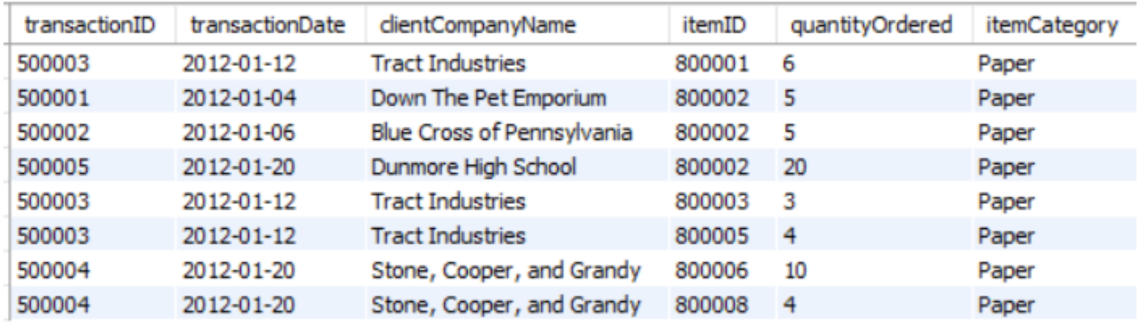
Use Case Realization:

```
SELECT t.transactionID, t.transactionDate, c.clientCompanyName,
     o.itemID, o.quantityOrdered, i.itemCategory
FROM dunderDB.Inventory i, dunderDB.ItemsOrdered o,
     dunderDB.Transactions t, dunderDB.Purchases p,
     dunderDB.Clients c
WHERE i.itemID = o.itemID AND o.transactionID = t.transactionID
     AND p.transactionID = t.transactionID
     AND p.clientID = c.clientID AND i.itemCategory = 'Paper';
```

Test Plan and Records:
INPUT: This use case requires the following input: item category of interest

EXPECTED: We expect all transactions that included the selected item category to be displayed to the user along with the transaction date, client name, item ID, and quantity ordered

ACTUAL:



transactionID	transactionDate	clientCompanyName	itemID	quantityOrdered	itemCategory
500003	2012-01-12	Tract Industries	800001	6	Paper
500001	2012-01-04	Down The Pet Emporium	800002	5	Paper
500002	2012-01-06	Blue Cross of Pennsylvania	800002	5	Paper
500005	2012-01-20	Dunmore High School	800002	20	Paper
500003	2012-01-12	Tract Industries	800003	3	Paper
500003	2012-01-12	Tract Industries	800005	4	Paper
500004	2012-01-20	Stone, Cooper, and Grandy	800006	10	Paper
500004	2012-01-20	Stone, Cooper, and Grandy	800008	4	Paper

Use Case: Delete items from inventory

Actor: Warehouse staff
Steps:

1. Actor clicks the button labeled “Permanently delete item from inventory”;
2. Prompt asks for item ID number;
3. Actor enters item ID number;
4. All information for the referenced client is displayed;
5. A message is displayed asking for the user to confirm that the item will be deleted from the database;
6. User clicks the button labeled “Confirm”

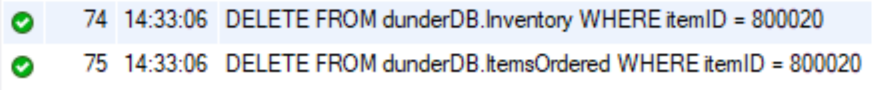
Use Case Realization:

```
DELETE FROM dunderDB.Inventory WHERE itemID = 800020;
DELETE FROM dunderDB.ItemsOrdered WHERE itemID = 800020;
```

Test Plan and Records:
INPUT: This use case requires the following input: item ID number

EXPECTED: We expect to see the specified item’s information deleted from the table, and unavailable to the user

ACTUAL:



```
✓  74  14:33:06  DELETE FROM dunderDB.Inventory WHERE itemID = 800020
✓  75  14:33:06  DELETE FROM dunderDB.ItemsOrdered WHERE itemID = 800020
```

Use Case: Add new sales transaction

Actor: Sales staff
Steps:

1. Actor clicks the button labeled “New sales transaction”;
2. A new transaction ID number is generated and displayed;
3. The actor enters the client ID, staff ID, transaction date, and transaction total cost;
4. If applicable, the actor can enter the date payment was received and the date the order was shipped;
5. A prompt is displayed that allows the user to click on individual items that were ordered or enter the item ID;
6. The actor selects the individual items that were ordered;
7. A new ID is generated for each item that was ordered;
8. The actor enters the quantity and the price for each item that was ordered;
9. All information is displayed back to the user;
10. A message is displayed asking for the user to confirm the order details;
11. User clicks the button labeled “Confirm”;
12. In the inventory data table, the item quantities are updated to subtract out the quantity of this new order

Use Case Realization:

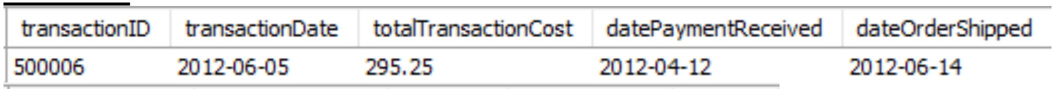
```
INSERT INTO dunderDB.Transactions (transactionID, transactionDate,
     totalTransactionCost, datePaymentReceived, dateOrderShipped)
VALUES(500006, '2012-06-05', 295.25, '2012-04-12', '2012-06-14');
INSERT INTO dunderDB.Purchases (transactionID, clientID, staffID)
VALUES(500006, 400030, 100177);
INSERT INTO dunderDB.ItemsOrdered (itemsOrderedID, quantityOrdered,
     itemPrice, transactionID, itemID)
VALUES(600016, 3, 70.5, 500006, 800002);
INSERT INTO dunderDB.ItemsOrdered (itemsOrderedID, quantityOrdered,
     itemPrice, transactionID, itemID)
VALUES(600017, 1, 83.75, 500006, 800004);
SELECT * FROM dunderDB.Transactions WHERE transactionID = 500006;
SELECT * FROM dunderDB.ItemsOrdered WHERE transactionID = 500006;
```

Test Plan and Records:

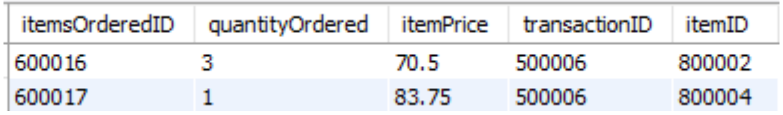
INPUT: This use case requires the following inputs:{ transactionID, transactionDate, totalTransactionCost, datePaymentRecieved, dateOrderShipped}

EXPECTED: We expect to see a new transaction in the Transactions table and displayed to the user

ACTUAL:



transactionID	transactionDate	totalTransactionCost	datePaymentReceived	dateOrderShipped
500006	2012-06-05	295.25	2012-04-12	2012-06-14



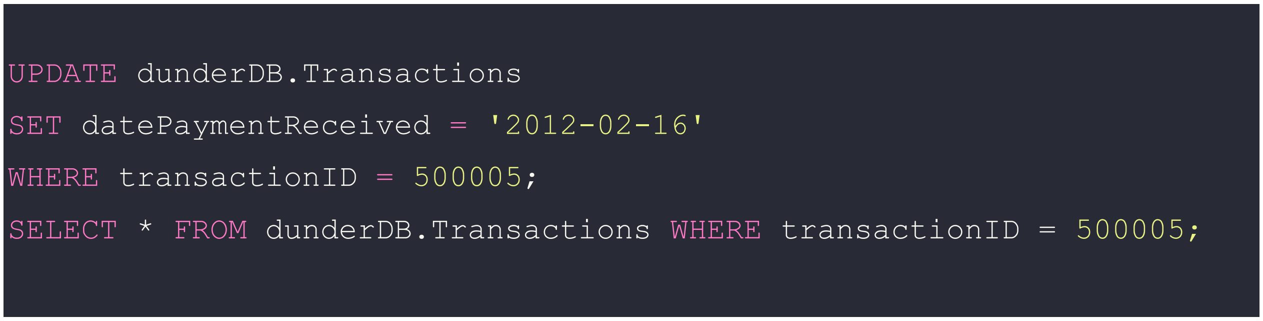
itemsOrderedID	quantityOrdered	itemPrice	transactionID	itemID
600016	3	70.5	500006	800002
600017	1	83.75	500006	800004

Use Case: Update transaction payment date

Actor: Accounting staff
Steps:

1. Actor clicks the button labeled “Update payment date for a transaction”;
2. Prompt asks for transaction ID number;
3. Actor enters transaction ID number;
4. All information for the referenced transaction is displayed;
5. Actor edits the date of payment received;
6. A message is displayed asking for the user to confirm the date of payment;
7. User clicks the button labeled “Confirm”

Use Case Realization:

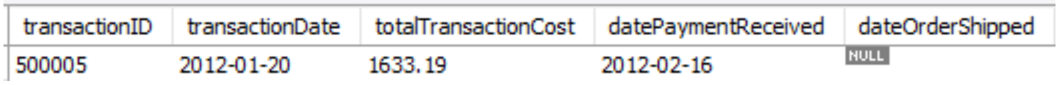


```
UPDATE dunderDB.Transactions
SET datePaymentReceived = '2012-02-16'
WHERE transactionID = 500005;
SELECT * FROM dunderDB.Transactions WHERE transactionID = 500005;
```

Test Plan and Records:
INPUT: This use case requires the following input: transaction ID number and the updated payment date

EXPECTED: We expect to see the specified transaction’s payment details updated and displayed to the user

ACTUAL:



transactionID	transactionDate	totalTransactionCost	datePaymentReceived	dateOrderShipped
500005	2012-01-20	1633.19	2012-02-16	NULL

Use Case: Update transaction shipping date

Actor: Warehouse staff
Steps:

1. Actor clicks the button labeled "Update shipping date for a transaction";
2. Prompt asks for transaction ID number;
3. Actor enters transaction ID number;
4. All information for the referenced transaction is displayed;
5. Actor edits the date of shipping for the transaction;
6. A message is displayed asking for the user to confirm the date of shipping;
7. User clicks the button labeled "Confirm"

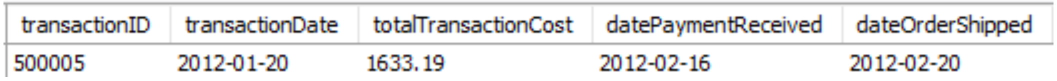
Use Case Realization:

```
UPDATE dunderDB.Transactions
SET dateOrderShipped = '2012-02-20'
WHERE transactionID = 500005;
SELECT * FROM dunderDB.Transactions WHERE transactionID = 500005;
```

Test Plan and Records:
INPUT: This use case requires the following input: transaction ID number and the updated shipment date

EXPECTED: We expect to see the specified transaction's shipping information updated and displayed to the user

ACTUAL:



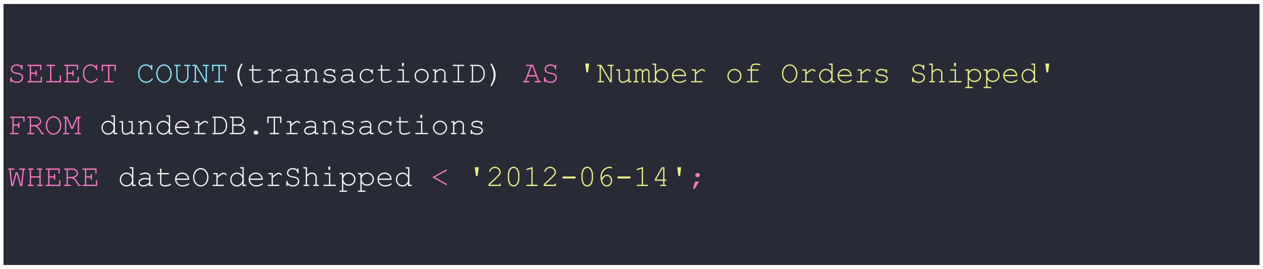
transactionID	transactionDate	totalTransactionCost	datePaymentReceived	dateOrderShipped
500005	2012-01-20	1633.19	2012-02-16	2012-02-20

Use Case: Count number of orders shipped before a specific date

Actor: Management, sales, accounting, or warehouse staff
Steps:

1. Actor clicks the button labeled “Number of orders shipped before a date”;
2. Prompt asks for the date of interest;
3. Actor enters date that they want used for calculating the total number of orders that were shipped prior to;
4. Database calculates the number of orders that were shipped before the target date;
5. The result is displayed to the user

Use Case Realization:

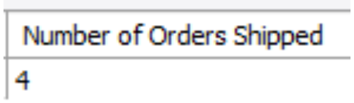


```
SELECT COUNT(transactionID) AS 'Number of Orders Shipped'
FROM dunderDB.Transactions
WHERE dateOrderShipped < '2012-06-14';
```

Test Plan and Records:
INPUT: This use case requires the following input: date of interest

EXPECTED: We expect to see the current count of orders shipped before the specified date displayed to the user

ACTUAL:



Number of Orders Shipped
4

Use Case: Display quantity of items ordered

Actor: Management, sales, accounting, or warehouse staff
Steps:

4. Actor clicks the button labeled “Display quantity of items ordered across all transactions”;
5. Database joins the transaction details with their related items ordered details;
6. The result is displayed to the user

Use Case Realization:

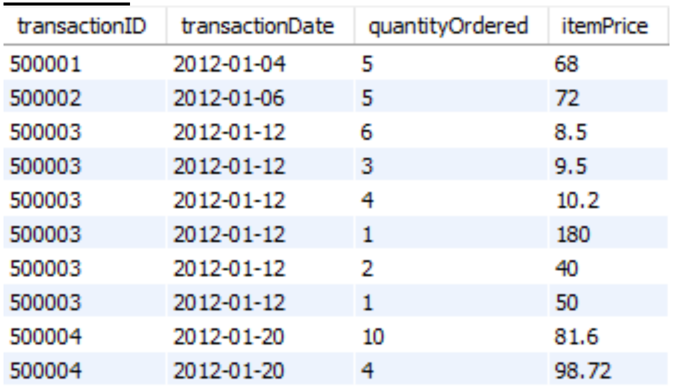
```
SELECT Transactions.transactionID, Transactions.transactionDate,
     ItemsOrdered.quantityOrdered, ItemsOrdered.itemPrice
FROM dunderDB.Transactions
LEFT JOIN ItemsOrdered
ON Transactions.transactionID = ItemsOrdered.transactionID
LIMIT 10;
```

Note: This code is limited to show 10 results for the purposes of this report.

Test Plan and Records:
INPUT: None, Selection only

EXPECTED: We expect a table with all items and their respective quantities displayed to the user

ACTUAL:



transactionID	transactionDate	quantityOrdered	itemPrice
500001	2012-01-04	5	68
500002	2012-01-06	5	72
500003	2012-01-12	6	8.5
500003	2012-01-12	3	9.5
500003	2012-01-12	4	10.2
500003	2012-01-12	1	180
500003	2012-01-12	2	40
500003	2012-01-12	1	50
500004	2012-01-20	10	81.6
500004	2012-01-20	4	98.72

Use Case: Delete a transaction

Actor: Sales staff
Steps:

1. Actor clicks the button labeled “Permanently delete a transaction”;
2. Prompt asks for transaction ID number;
3. Actor enters transaction ID number;
4. All information for the referenced transaction is displayed;
5. A message is displayed asking for the user to confirm that this will be deleted from the database;
6. User clicks the button labeled “Confirm”

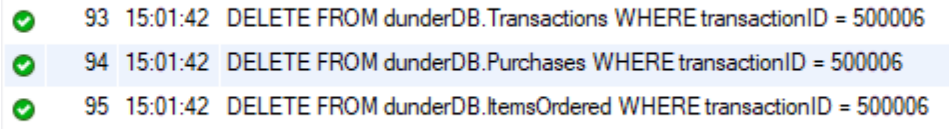
Use Case Realization:

```
DELETE FROM dunderDB.Transactions WHERE transactionID = 500006;
DELETE FROM dunderDB.Purchases WHERE transactionID = 500006;
DELETE FROM dunderDB.ItemsOrdered WHERE transactionID = 500006;
```

Test Plan and Records:
INPUT: This use case requires the following input: transaction ID number

EXPECTED: We expect the specified transaction and all related information deleted from the database

ACTUAL:



```
✓  93  15:01:42  DELETE FROM dunderDB.Transactions WHERE transactionID = 500006
✓  94  15:01:42  DELETE FROM dunderDB.Purchases WHERE transactionID = 500006
✓  95  15:01:42  DELETE FROM dunderDB.ItemsOrdered WHERE transactionID = 500006
```

Use Case: Add new items to a transaction

Actor: Sales staff
Steps:

1. Actor clicks the button labeled “Add new items to an existing transaction”;
2. A new items ordered ID number is generated and displayed;
3. The actor enters the transaction ID that the item will be added to;
4. A prompt is displayed that allows the user to click on individual items that will be added to the transaction or enter the item ID or item name;
5. The actor selects the individual items to be added to the transaction;
6. The actor enters the quantity and the price for each item that was ordered;
7. All information is displayed back to the user;
8. A message is displayed asking for the user to confirm the order details;
9. User clicks the button labeled “Confirm”;
10. In the inventory data table, the item quantities are updated to subtract out the quantity of this new order

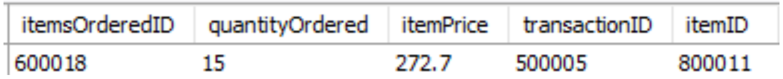
Use Case Realization:

```
INSERT INTO dunderDB.ItemsOrdered (itemsOrderedID, quantityOrdered,
     itemPrice, transactionID, itemID)
VALUES(600018, 15, 272.70, 500005, 800011);
SELECT * FROM dunderDB.ItemsOrdered WHERE itemsOrderedID = 600018;
```

Test Plan and Records:
INPUT: This use case requires the following inputs: itemsOrderedID, quantityOrdered, itemPrice, transactionID, itemID

EXPECTED: We expect to see a new item added to the existing transaction and displayed to the user

ACTUAL:



itemsOrderedID	quantityOrdered	itemPrice	transactionID	itemID
600018	15	272.7	500005	800011

Use Case: Update the quantity of an item in an order

Actor: Sales staff
Steps:

1. Actor clicks the button labeled “Update quantity of items in a transaction”;
2. Prompt asks for item ordered ID number;
3. Actor enters item ordered ID number;
4. All items ordered for the referenced transaction are displayed;
5. Actor edits the quantity of a items as needed;
6. A message is displayed asking for the user to confirm the changes;
7. User clicks the button labeled “Confirm”

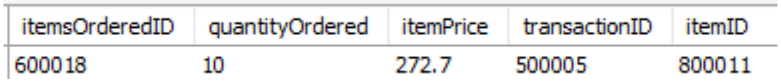
Use Case Realization:

```
UPDATE dunderDB.ItemsOrdered
SET quantityOrdered = 10
WHERE itemsOrderedID = 600018;
SELECT * FROM dunderDB.ItemsOrdered WHERE itemsOrderedID = 600018;
```

Test Plan and Records:
INPUT: This use case requires the following input: item ordered ID number and the updated item quantity

EXPECTED: We expect to see the specified transaction’s information updated and displayed to the user

ACTUAL:



itemsOrderedID	quantityOrdered	itemPrice	transactionID	itemID
600018	10	272.7	500005	800011

Use Case: Update the price of an item in an order

Actor: Sales staff
Steps:

1. Actor clicks the button labeled “Update price of items in a transaction”;
2. Prompt asks for item ordered ID number or client name and date of transaction;
3. Actor enters item ordered ID number or client name and date of transaction;
4. All items ordered for the referenced transaction are displayed;
5. Actor edits the price of a items as needed;
6. A message is displayed asking for the user to confirm the changes;
7. User clicks the button labeled “Confirm”

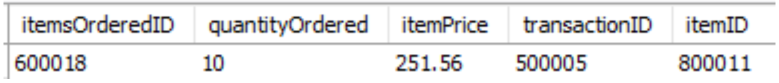
Use Case Realization:

```
UPDATE dunderDB.ItemsOrdered
SET itemPrice = 251.56
WHERE itemsOrderedID = 600018;
SELECT * FROM dunderDB.ItemsOrdered WHERE itemsOrderedID = 600018;
```

Test Plan and Records:
INPUT: This use case requires the following input: item ordered ID number and the updated item price

EXPECTED: We expect to see the specified transaction’s information updated and displayed to the user

ACTUAL:



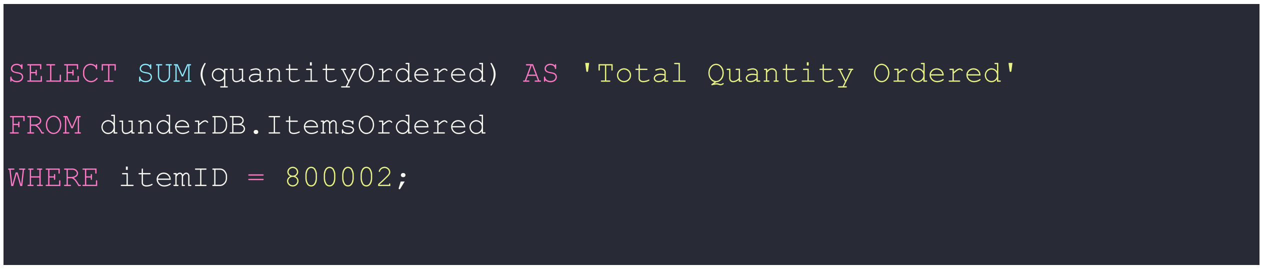
itemsOrderedID	quantityOrdered	itemPrice	transactionID	itemID
600018	10	251.56	500005	800011

Use Case: Display the total sales quantity of an item

Actor: Management, sales, accounting, or warehouse staff
Steps:

1. Actor clicks the button labeled “Display total sales quantity of an item”;
2. Prompt asks for the item of interest;
3. Actor enters item ID or item name that they want used for calculating the total quantity of sales for;
4. Database calculates the total quantity of sales for that item;
5. The result is displayed to the user

Use Case Realization:

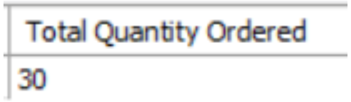


```
SELECT SUM(quantityOrdered) AS 'Total Quantity Ordered'
FROM dunderDB.ItemsOrdered
WHERE itemID = 800002;
```

Test Plan and Records:
INPUT: This use case requires the following input: item ID

EXPECTED: We expect the total quantity of orders for the specified item displayed to the user

ACTUAL:



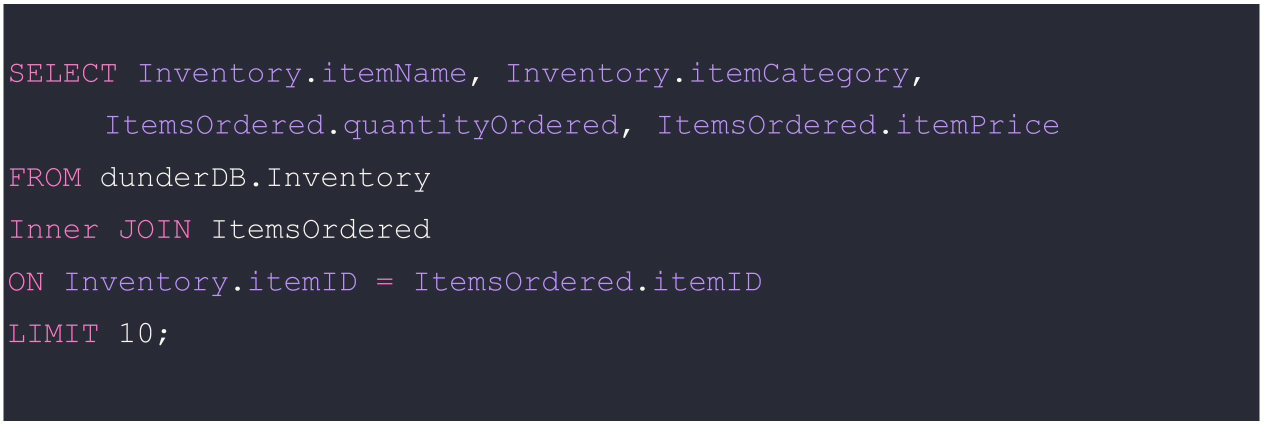
Total Quantity Ordered
30

Use Case: Display all inventory that has been ordered

Actor: Management, sales, accounting, or warehouse staff
Steps:

1. Actor clicks the button labeled “Display details on all inventory that has been ordered”;
2. Database joins the inventory details with their related items ordered details;
3. The result is displayed to the user

Use Case Realization:



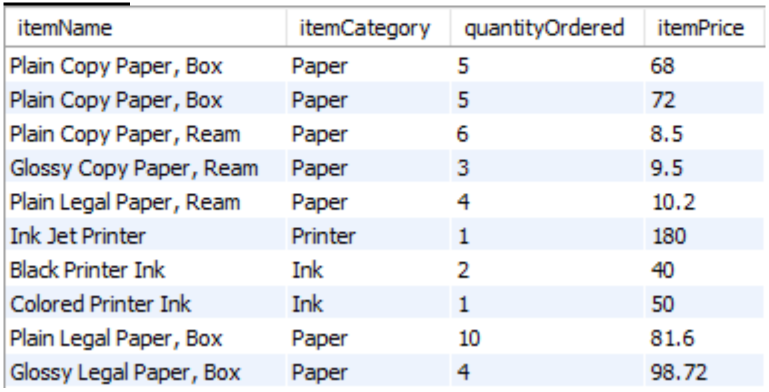
```
SELECT Inventory.itemName, Inventory.itemCategory,
     ItemsOrdered.quantityOrdered, ItemsOrdered.itemPrice
FROM dunderDB.Inventory
Inner JOIN ItemsOrdered
ON Inventory.itemID = ItemsOrdered.itemID
LIMIT 10;
```

Note: This code is limited to show 10 results for the purposes of this report.

Test Plan and Records:
INPUT: None, Selection only

EXPECTED: We expect a table with all inventory ordered in a transaction displayed to the user

ACTUAL:



itemName	itemCategory	quantityOrdered	itemPrice
Plain Copy Paper, Box	Paper	5	68
Plain Copy Paper, Box	Paper	5	72
Plain Copy Paper, Ream	Paper	6	8.5
Glossy Copy Paper, Ream	Paper	3	9.5
Plain Legal Paper, Ream	Paper	4	10.2
Ink Jet Printer	Printer	1	180
Black Printer Ink	Ink	2	40
Colored Printer Ink	Ink	1	50
Plain Legal Paper, Box	Paper	10	81.6
Glossy Legal Paper, Box	Paper	4	98.72

Use Case: Delete an item from a transaction

Actor: Sales staff
Steps:

1. Actor clicks the button labeled “Permanently delete an item from a transaction”;
2. Prompt asks for item ordered ID number;
3. Actor enters item ordered ID number;
4. All information for the referenced item ordered is displayed;
5. A message is displayed asking for the user to confirm that this will be deleted from the database;
6. User clicks the button labeled “Confirm”

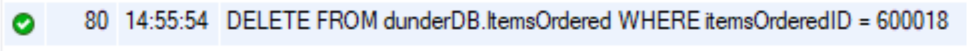
Use Case Realization:

```
DELETE FROM dunderDB.ItemsOrdered
WHERE itemsOrderedID = 600018;
```

Test Plan and Records:
INPUT: This use case requires the following input: item ordered ID number

EXPECTED: We expect the specified item ordered in transaction to be deleted from the database

ACTUAL:



```
80	14:55:54	DELETE FROM dunderDB.ItemsOrdered WHERE itemsOrderedID = 600018
```

Conclusion

The team originally set out to develop a functional database and management system for the primary use and modeling of the Scranton, PA Dunder Mifflin Paper Company. What the team has accomplished is precisely that. We have developed an easily accessible, user-friendly, working, functional database. This database allows the company to store, modify, and access different aspects of the company, such as: information about staff, emergency contact information, HR incidents, clients, transactions, inventory, and items purchased. The database was normalized to the third normal form to avoid issues with data redundancy and update anomalies. Relationships were established between the entities identified during the conceptual database design phase. With few errors to correct, the development process was relatively smooth, and the working database is successfully being hosted on Amazon Web Services (AWS). Time efficiency and proper allocation of manpower to cover the different phases and tasks were integral to the success of this project.

References

Below is a link to the demonstration video for this project:
https://drive.google.com/file/d/1zG0LvqUk7GkCP2dmWa7F56SlZ7x4nuaG/view?usp=sharing

Data for staff members was taken from the following sources:
https://theoffice.fandom.com/wiki/Main_Page
https://en.wikipedia.org/wiki/List_of_The_Office_(American_TV_series)_characters

Estimates for staff salaries was taken from the following source:
https://screenrant.com/the-office-character-salaries-dunder-mifflin-reddit/

Data for HR incidents was taken from the following source:
https://theoffice.fandom.com/wiki/List_of_Jim%27s_pranks

Data for clients was taken from the following source:
https://theoffice.fandom.com/wiki/Clients_of_Dunder_Mifflin

Phone numbers, addresses, dates, and names that were not available in the sources listed above were generated using fictitious data.

Submission confirmation: **a1fd0da3-90da-4cab-97f4-b5ec7ec470fd**