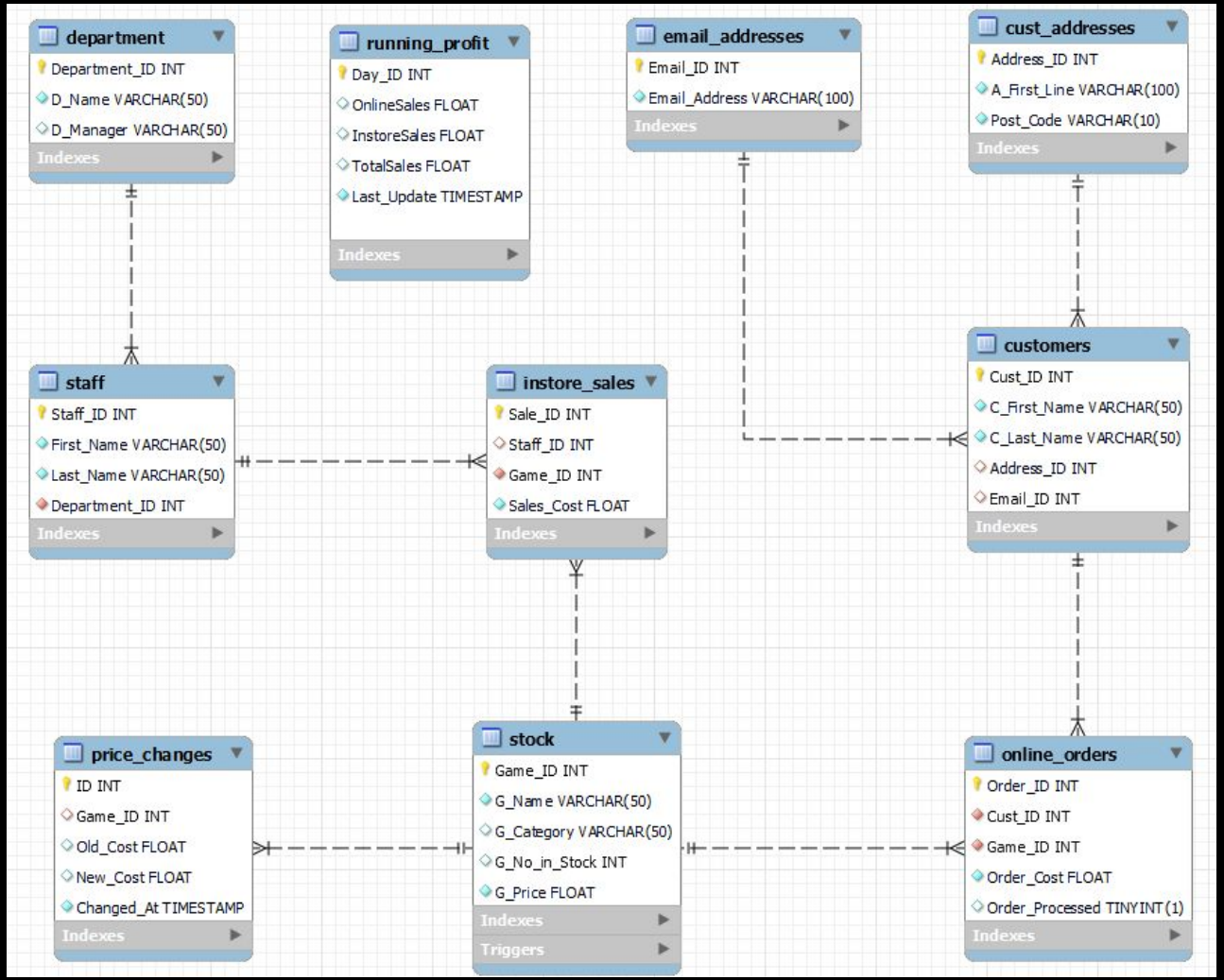
INTRO TO DATA AND SQL PROJECT:

BOARD GAME SHOP DATABASE

REBECCA ROSCOE

www.linkedin.com/in/rebecca-roscoe-4016a2289



For my project, I created a database on MySQL based on the idea of a Board Game shop. In order to do so, I designed and created tables to capture data about the shops staff, customers, stock and sales.

These were then used to perform a variety of the SQL queries learnt during the Intro to Data and SQL course.

CREATING TABLES

```
CREATE TABLE staff
(Staff_ID INT NOT NULL PRIMARY KEY,
First_Name VARCHAR (50) NOT NULL,
Last_Name VARCHAR (50) NOT NULL,
Department_ID INT NOT NULL
);

CREATE TABLE customers
(Cust_ID INT NOT NULL PRIMARY KEY,
C_First_Name VARCHAR (50) NOT NULL,
C_Last_Name VARCHAR (50) NOT NULL,
Address_ID INT NULL,
Email_ID INT NULL
);
```

```
CREATE TABLE online_orders
(Order_ID INT NOT NULL PRIMARY KEY,
Cust_ID INT NOT NULL,
Game_ID INT NOT NULL,
Order_Cost FLOAT NOT NULL,
Order_Processed BOOLEAN,
FOREIGN KEY (Cust_ID) REFERENCES customers(Cust_ID) ON UPDATE CASCADE,
FOREIGN KEY (Game_ID) REFERENCES stock(Game_ID) ON UPDATE CASCADE
);
```

Due to the nature of my database, additional foreign keys were added after the creation of further tables.

```
ALTER TABLE staff
ADD FOREIGN KEY (Department_ID) REFERENCES department(Department_ID) ON UPDATE CASCADE;

ALTER TABLE customers
ADD FOREIGN KEY (Address_ID) REFERENCES cust_addresses(Address_ID) ON UPDATE CASCADE,
ADD FOREIGN KEY (Email_ID) REFERENCES email_addresses(Email_ID) ON UPDATE CASCADE;
```

INSERTING DATA

I manually inputted the data into each of the tables, using 'INSERT INTO'

```
INSERT INTO department
(Department_ID, D_Name, D_Manager)
VALUES
(001, 'Sales', 'Mike'),
(002, 'Warehouse', 'Dave'),
(003, 'Purchasing', 'Helen'),
(004, 'HR', 'Sam');
```

```
INSERT INTO customers
(Cust_ID, C_First_Name, C_Last_Name, Address_ID, Email_ID)
VALUES
(001, 'Robert', 'Smith', 001, 001),
(002, 'Jerry', 'Evans', 002, NULL),
(003, 'Debra', 'Cook', 003, 002),
(004, 'Dorothy', 'Morris', 004, 003),
(005, 'Joesph', 'Morris', 004, 004),
```

```
INSERT INTO stock
(Game_ID, G_Name, G_Category, G_No_in_Stock, G_Price)
VALUES
(001, 'Gloomhaven', 'Adventure', 4, 126.99),
(002, 'Pandemic Legacy: Season 1', 'Cooperative', 10, 64.99),
(003, 'Brass: Birmingham', 'Economic Strategy', 11, 59.99),
(004, 'Terraforming Mars', 'Economic Strategy', 7, 54.99),
(005, 'Gloomhaven: Jaws of the Line', 'Adventure', 8, 39.99),
(006, 'Gaia Project', 'Strategy', 11, 66.99),
(007, 'Star Wars: Rebellion', 'Strategy', 3, 80.99),
(008, 'Twilight Struggle', 'Historical Strategy', 0, 69.99),
(009, 'Great Western Trail 2nd Edition', 'Economic Strategy', 12, 35.99),
(010, 'Spirit Island', 'Cooperative', 3, 69.99),
(011, 'Scythe', 'Historical Strategy', 7, 66.99),
(012, 'Terra Mystica', 'Strategy', 9, 74.99),
(013, '7 Wonders Duel', 'Engine-Building', 13, 18.99),
(014, 'Brass: Lancashire', 'Economic Strategy', 9, 58.99),
(015, 'Wingspan', 'Engine-Building', 8, 43.99),
(016, 'A Feast for Odin', 'Worker Placement', 5, 89.99),
(017, 'Viticulture', 'Worker Placement', 3, 26.99),
(018, 'Root', 'Worker Placement', 6, 43.99),
```

EXAMPLE TABLES

Staff_ID	First_Name	Last_Name	Department_ID
37	Dave	Woods	2
42	Mike	Sanderson	1
47	Sam	Holland	4
58	Sonia	Bonner	2
73	Helen	Cannon	3
74	Alfred	Castaneda	2
75	Kimberley	Potter	4
76	Hari	Foley	2
82	Nathanael	Bonner	3
83	Saffron	Cameron	2
86	Tatiana	Knowles	2
87	Natalie	Pierce	1
92	Amir	Hutchinson	1
93	Frankie	Walls	2
96	Aiza	Duke	1
99	Allan	Horton	1
102	Theresa	Spencer	4
103	Harris	David	1
105	Shannon	Lindsay	2
106	Alanna	Collins	1
108	Kareem	Obrien	1

Cust_ID	C_First_Name	C_Last_Name	Address_ID	Email_ID
1	Robert	Smith	1	1
2	Jerry	Evans	2	NULL
3	Debra	Cook	3	2
4	Dorothy	Morris	4	3
5	Joesph	Morris	4	4
6	Grace	Harrison	5	5
7	Thomas	Roberts	5	6
8	Paul	Young	6	7
9	Alexander	Parker	7	8
10	Alexander	Anderson	9	NULL
11	Pamela	Parker	7	8
12	Adam	Hughes	10	9
13	Justin	Kelly	NULL	10

Order_ID	Cust_ID	Game_ID	Order_Cost	Order_Processed
101	15	29	38.99	1
102	15	15	43.99	1
103	11	20	48.99	1
104	1	32	17.99	1
105	3	1	126.99	1
106	7	27	39.99	1
107	4	12	74.99	1
108	6	26	48.99	1
109	9	18	43.99	1
110	12	26	48.99	1
111	7	23	25.99	1
112	20	34	29.99	1
113	5	14	58.99	1
114	12	26	48.99	1

Sale_ID	Staff_ID	Game_ID	Sales_Cost
501	87	15	43.99
502	87	23	25.99
503	92	11	66.99
504	87	26	48.99
505	92	23	25.99
506	92	20	48.99
507	92	26	48.99
508	96	2	64.99
509	99	15	43.99
510	99	28	48.99
511	96	31	17.99
512	87	18	43.99
513	96	25	50.99
514	96	2	64.99
515	87	15	43.99
516	99	21	38.99
517	99	26	48.99
518	99	34	29.99

STORED FUNCTION

I created a stored function that tells you the department of a staff member when given their department ID.

```
DELIMITER //
CREATE FUNCTION Staff_Department(Department_ID INT)
RETURNS VARCHAR(50)
DETERMINISTIC
BEGIN
    DECLARE staff_department VARCHAR(50);
    IF Department_ID = 001 THEN
        SET staff_department = 'Sales';
    ELSEIF Department_ID = 002 THEN
        SET staff_department = 'Warehouse';
    ELSEIF Department_ID = 003 THEN
        SET staff_department = 'Purchasing';
    ELSEIF Department_ID = 004 THEN
        SET staff_department = 'HR';
    END IF;
RETURN (staff_department);
END //
DELIMITER ;
```

```
SELECT *, Staff_Department(Department_ID) AS Department
FROM staff;
```

Staff_ID	First_Name	Last_Name	Department_ID	Department
37	Dave	Woods	2	Warehouse
42	Mike	Sanderson	1	Sales
47	Sam	Holland	4	HR
58	Sonia	Bonner	2	Warehouse
73	Helen	Cannon	3	Purchasing
74	Alfred	Castaneda	2	Warehouse
75	Kimberley	Potter	4	HR
76	Hari	Foley	2	Warehouse
82	Nathanael	Bonner	3	Purchasing
83	Saffron	Cameron	2	Warehouse
86	Tatiana	Knowles	2	Warehouse
87	Natalie	Pierce	1	Sales
92	Amir	Hutchinson	1	Sales
93	Frankie	Walls	2	Warehouse
96	Aiza	Duke	1	Sales
99	Allan	Horton	1	Sales
102	Theresa	Spencer	4	HR
103	Harris	David	1	Sales
105	Shannon	Lindsay	2	Warehouse
106	Alanna	Collins	1	Sales
108	Kareem	Obrien	1	Sales

STORED PROCEDURE

I created a stored procedure that allows you to easily input new sales into the instore_sales table.

```
DELIMITER //
CREATE PROCEDURE InsertSales(IN Sale_ID INT, IN Staff_ID INT, Game_ID INT, Sales_Cost FLOAT)
BEGIN
    INSERT INTO instore_sales(Sale_ID, Staff_ID, Game_ID, Sales_Cost)
    VALUES (Sale_ID, Staff_ID, Game_ID, Sales_Cost);
END //
DELIMITER ;

CALL InsertSales(531, 92, 31, 17.99);

SELECT * FROM instore_sales;
```

Sale_ID	Staff_ID	Game_ID	Sales_Cost
505	92	23	25.99
506	92	20	48.99
507	92	26	48.99
508	96	2	64.99
509	99	15	43.99
510	99	28	48.99
511	96	31	17.99
512	87	18	43.99
513	96	25	50.99
514	96	2	64.99
515	87	15	43.99
516	99	21	38.99
517	99	26	48.99
518	99	34	29.99
519	92	26	48.99
520	92	9	35.99
521	96	21	38.99
522	103	15	43.99
523	92	17	26.99
524	103	24	49.99
525	106	31	17.99
526	108	26	48.99
527	99	16	89.99
528	99	14	58.99
529	106	28	48.99
531	92	31	17.99

TRIGGER

I created a trigger that kept track of game price changes, by updating the price_changes table.

```
DELIMITER //
CREATE TRIGGER PriceChange
AFTER UPDATE ON stock
FOR EACH ROW
BEGIN
    IF OLD.G_Price <> NEW.G_Price THEN
        INSERT INTO price_changes(Game_ID, Old_Cost, New_Cost)
        VALUES (old.Game_ID, old.G_Price, new.G_Price);
    END IF;
END //
DELIMITER ;

UPDATE stock
SET G_Price = 39.99
WHERE Game_ID = 018;

SELECT * FROM price_changes;
```

ID	Game_ID	Old_Cost	New_Cost	Changed_At
1	12	74.99	76.99	2023-08-31 15:11:35
2	2	64.99	60.99	2023-08-31 15:12:14
3	12	76.99	41.99	2023-08-31 17:02:58
4	12	41.99	76.99	2023-08-31 17:03:17
5	18	43.99	39.99	2023-09-07 14:38:34

```
CREATE TABLE Price_Changes
(ID INT AUTO_INCREMENT PRIMARY KEY,
Game_ID INT,
Old_Cost FLOAT,
New_Cost FLOAT,
Changed_At TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

EVENT

I created an event that updates the running_profit once a day with the total instore sales, online sales and combined sales.

```
DELIMITER //
CREATE EVENT RunningProfit
    ON SCHEDULE EVERY 24 HOUR
DO BEGIN
    INSERT INTO running_profit(OnlineSales, InstoreSales, TotalSales)
        SELECT SUM(Order_Cost) AS OnlineSales, SUM(Sales_Cost) AS InstoreSales, SUM(Order_Cost) + SUM(Sales_Cost) AS TotalSales
        FROM online_orders, instore_sales;
END //
DELIMITER ;

SELECT * FROM running_profit;
```

	Day_ID	OnlineSales	InstoreSales	TotalSales	Last_Update
▶	1	37199.2	35578.2	72777.3	2023-08-31 17:01:29

```
CREATE TABLE Running_Profit
(Day_ID INT AUTO_INCREMENT PRIMARY KEY,
OnlineSales FLOAT,
InstoreSales FLOAT,
TotalSales FLOAT,
Last_Update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

DATA ANALYSIS

SUBQUERIES

I created a query with subqueries to SELECT the First Name and Last name of each customer who had placed an online order for a ‘worker placement’ game.

```
SELECT C_First_Name, C_Last_Name
FROM customers
WHERE Cust_ID IN (SELECT DISTINCT Cust_ID
                  FROM online_orders
                  WHERE Game_ID IN (SELECT Game_ID
                                    FROM stock
                                    WHERE G_Category = 'Worker Placement'));
```

C_First_Name	C_Last_Name
Alexander	Parker
Isabella	Morgan
Donald	Collins

JOINS AND VIEWS

I created a VIEW to show the Sales ID, Staff Name, Game Name and Cost of each instore order. This was done using LEFT JOIN to combine 3 tables.

```
CREATE VIEW vw_instore AS
SELECT i.Sale_ID, CONCAT(s.First_Name, ' ', s.Last_Name) AS Name, g.G_Name, i.sales_cost
FROM instore_sales i
LEFT JOIN staff s
ON i.Staff_ID = s.Staff_ID
LEFT JOIN stock g
ON i.Game_ID = g.Game_ID;

SELECT * FROM vw_instore;
```

Sale_ID	Name	G_Name	sales_cost
501	Natalie Pierce	Wingspan	43.99
502	Natalie Pierce	Azul	25.99
503	Amir Hutchinson	Scythe	66.99
504	Natalie Pierce	Clank!: A Deck-Building Adventure	48.99
505	Amir Hutchinson	Azul	25.99
506	Amir Hutchinson	Agricola	48.99
507	Amir Hutchinson	Clank!: A Deck-Building Adventure	48.99
508	Aiza Duke	Pandemic Legacy: Season 1	64.99
509	Allan Horton	Wingspan	43.99
510	Allan Horton	Five Tribes	48.99
511	Aiza Duke	Patchwork	17.99
512	Natalie Pierce	Root	43.99
513	Aiza Duke	Marvel Champions: The Card Game	50.99
514	Aiza Duke	Pandemic Legacy: Season 1	64.99
515	Natalie Pierce	Wingspan	43.99
516	Allan Horton	Everdell	38.99
517	Allan Horton	Clank!: A Deck-Building Adventure	48.99
518	Allan Horton	Pandemic	29.99
519	Amir Hutchinson	Clank!: A Deck-Building Adventure	48.99
520	Amir Hutchinson	Great Western Trail 2nd Edition	35.99
521	Aiza Duke	Everdell	38.99
522	Harris David	Wingspan	43.99
523	Amir Hutchinson	Viticulture	26.99
524	Harris David	Clank! in Space!: A Deck-Building ...	49.99

JOINS AND VIEWS

I created a second VIEW to show the Order ID, Customer Name, Customer Postcode, Game Name and Cost of each online order. This was done using LEFT JOIN to combine 4 tables.

```
CREATE VIEW vw_online AS
SELECT o.Order_ID, CONCAT(c.C_First_Name, ' ', c.C_Last_Name) AS Name, a.Post_Code, g.G_Name, o.Order_Cost
FROM online_orders o
LEFT JOIN customers c
ON o.Cust_ID = c.Cust_ID
LEFT JOIN cust_addresses a
ON c.Address_ID = a.Address_ID
LEFT JOIN stock g
ON o.Game_ID = g.Game_ID;

SELECT * FROM vw_online;
```

Order_ID	Name	Post_Code	G_Name	Order_Cost
101	Donald Collins	WC46 9IR	Lords of Waterdeep	38.99
102	Donald Collins	WC46 9IR	Wingspan	43.99
103	Pamela Parker	N87 9UP	Agricola	48.99
104	Robert Smith	NW60 9WY	Codenames	17.99
105	Debra Cook	E48 3DP	Gloomhaven	126.99
106	Thomas Roberts	E38 6VD	The Quacks of Quedlingburg	39.99
107	Dorothy Morris	W19 8FY	Terra Mystica	74.99
108	Grace Harrison	E38 6VD	Clank!: A Deck-Building Adventure	48.99
109	Alexander Parker	N87 9UP	Root	43.99
110	Adam Hughes	SW61 4NM	Clank!: A Deck-Building Adventure	48.99
111	Thomas Roberts	E38 6VD	Azul	25.99
112	Jose Jackson	W49 3YS	Pandemic	29.99
113	Joesph Morris	W19 8FY	Brass: Lancashire	58.99
114	Adam Hughes	SW61 4NM	Clank!: A Deck-Building Adventure	48.99
115	Betty Brown	N27 3VD	Scythe	66.99
116	Jonathan Morgan	SE60 8QC	Patchwork	17.99
117	Debra Cook	E48 3DP	Gloomhaven: Jaws of the Line	39.99

QUERYING VIEWS

I used the first view to find out the top 3 salespersons and the total SUM of their sales. This involved using GROUP BY, ORDER BY and LIMIT.

```
SELECT Name, SUM(Sales_Cost) AS Total_Sold
FROM vw_instore
GROUP BY Name
ORDER BY Total_Sold DESC
LIMIT 3;
```

Name	Total_Sold
Allan Horton	359.9300060272217
Amir Hutchinson	320.9200038909912
Aiza Duke	237.94999885559082

QUERYING VIEWS

I used the second view to find out the game name and number of online orders for each game sold for more than £40. This involved using GROUP BY and HAVING.

```
SELECT G_Name, Order_Cost, COUNT(Order_ID) AS No_Sold
FROM vw_online
GROUP BY G_Name, Order_Cost
HAVING Order_Cost > 40
ORDER BY No_Sold DESC;
```

G_Name	Order_Cost	No_Sold
Clank!: A Deck-Building Adventure	48.99	4
Wingspan	43.99	2
Agricola	48.99	1
Gloomhaven	126.99	1
Terra Mystica	74.99	1
Root	43.99	1
Brass: Lancashire	58.99	1
Scythe	66.99	1
Clank! in Space!: A Deck-Building Adventure	49.99	1
Twilight Struggle	69.99	1
Food Chain Magnate	77.99	1
Brass: Birmingham	59.99	1