CANDIDATE NUMBER (C-NUMBER)	C2096215
MODULE NAME	Data Design Management
WORD COUNT	---
SUBMISSION DATE	31/01/2024

DECLARATION

I certify that this assessment submission is entirely my work and I have fully referenced and correctly cited the work of others, where required. I also confirm the contents of my submission have not been generated by a third party, or through an Artificial Intelligence generative system*.

I have read the Student Discipline Regulations (Student Discipline Regulations) and understand any Assessment Related Offence/ Academic Misconduct may result penalties being applied.

By submitting this assessment submission, I am confirming that I am fit to sit according to the Assessment Regulations.

I declare that:

- This is my own unaided work.
  Yes ☒
  No ☐

- The word count stated by me is correct.
  Yes ☒
  No ☐

- I'm happy for my work to be retained on the Elite repository and made available to staff and future students**
  Yes ☒
  No ☐

*Please note that all the assignments are submitted to Turnitin.
**Please note personal information (such as names) will be deleted.

Data Design Assessment

Farzam Ferydooni

Student Number:0381952

(Lui M [Photograph] 2009)

Outline

- Requirements And Design Considerations
- Designing
- Implementation
- Data Entry
- Current State of Tables
- Assessment Tasks
- Use Cases
- Conclusion

Requirements And Design Considerations

Requirements

- Online shop
- Services: selling Books, Movies , Albums (Music Recordes)
- Location To service: World Wide
- Database Type: relational database, centralise, MySQL
- Environment for implementation: MySQL Workbench 8
- Design Environment: MS Visio
- Design: Chen ERD, Crow’s Foot database notation

Design Considerations

- A database is the foundation of every service, it is crucial to invest extreme effort in its design(Silberschatz et al. 1–25). It could help us to reach sustainable service. Some of main considerations are:
  - Performance and scalability: One of the main factors in online business, especially in the worldwide market, is providing high-performance service for various scale customers. This could be achieve by deferent layers from high-performance network and processing power to modular design in database and also redundancy in those layers and designs.
  - Security: Encryption, data masking, and access Control (with segregation of duty) could be some example of ways to provide security over the database. In order that was mentioned before in other layers, backup and access control by firewall could be considered.
  - Maintainability: Designing a service is not an on time job. But it needs a lot of refer, to repair or even redesign with regard to service developments based on service owner or customer demands. For this reason, Databases should be designed and documented to facilitate these need.

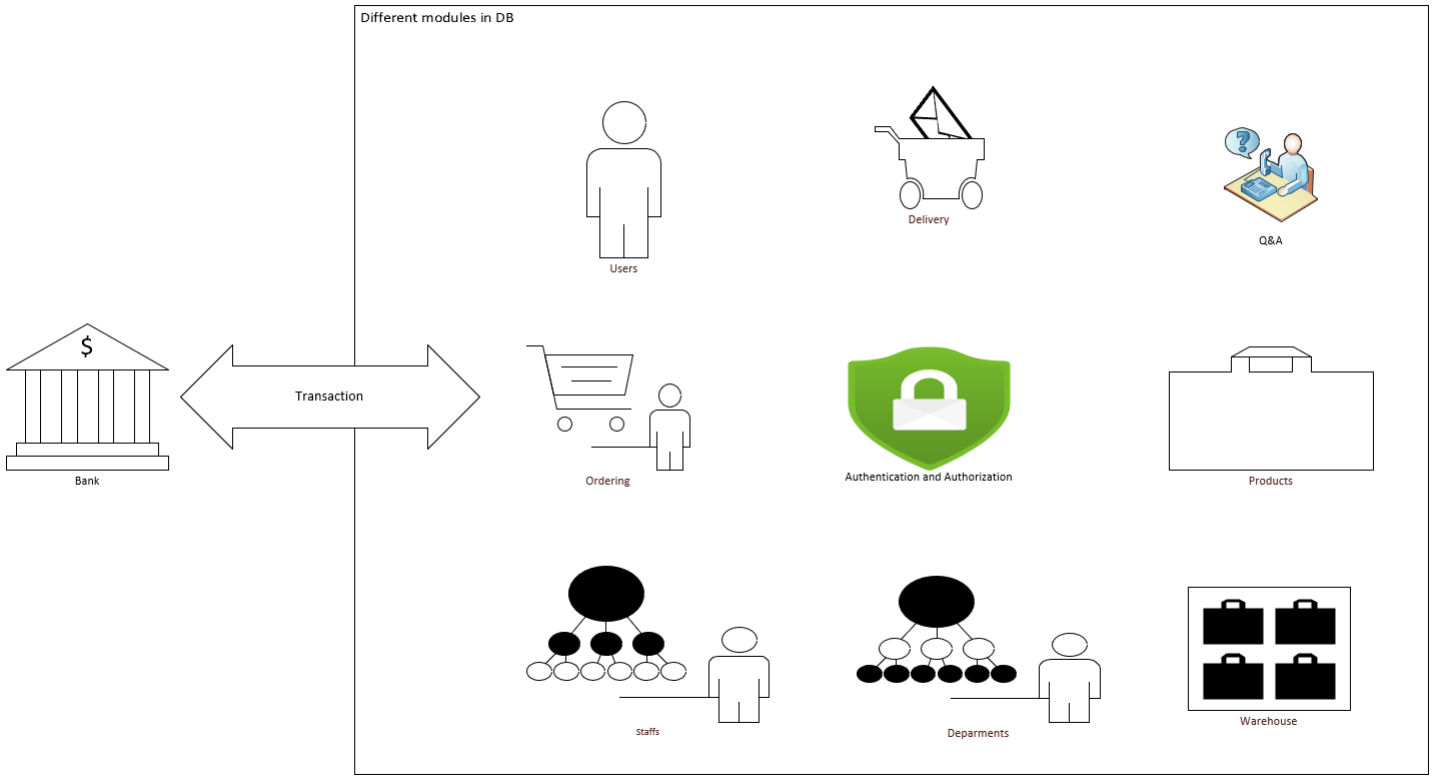
Lean Operation in Database design (Graupp, 2022)

- One systematic way to cover those demands is through of the Lean Operation. here are some practise for the lean design:
  - Value propositioning and targeting audience
  - Identifying unnecessary process in designing
  - Data Normalization
  - standardization in database design for example: Naming, indentation, documentation
  - Testing and validation procedures
  - Monitoring and evaluating

Lean Operation in Database design-benefits(Graupp, 2022)

- Because It is a systematic operation, Here are some benefit of using Lean operation in database design:
  - Increasing flexibility for fast and easy changes and updates
  - Increasing customer by covering their needs
  - Regular security and compliance control
  - Reduce cost of development with minimizing unnecessary features

Project-Macro View



In macro view, the project consist of external part (Bank and PSP for transaction) and internal part (Database and frontend and backend) and connectivity (private line and internet)

Project-Description

At least the following items should be considered in the programming of this project:

- User Section
  - Define users
  - Users activities logging
  - Review and rating
  - Shopping
- Employee section
  - Define employees
  - Assigning them to Department
  - Employee activities logging
  - Assigning to a task
- Departments
  - Define Departments
  - Staff management operation data

Project-Description

- Authentication and Authorization
  - Authentication and Authorization for users and staff
- Product
  - Define product
  - Product review and score
- Warehouse
  - All warehouse operation
- Ordering
  - Order listing
- Billing and Transaction
  - Bank payment and verification

Project-Description

- Financial Management
- Order Delivery
  - Delivery details
- Q&A and Review and rating
  - Every Q&A about products and services
- Feedback and complain centre
  - Submit customers and employees feedbacks and complains
  - Follow up feedbacks and complains
- Service Development
  - Improve service with users feedbacks

What is Crow's Foot Notation?

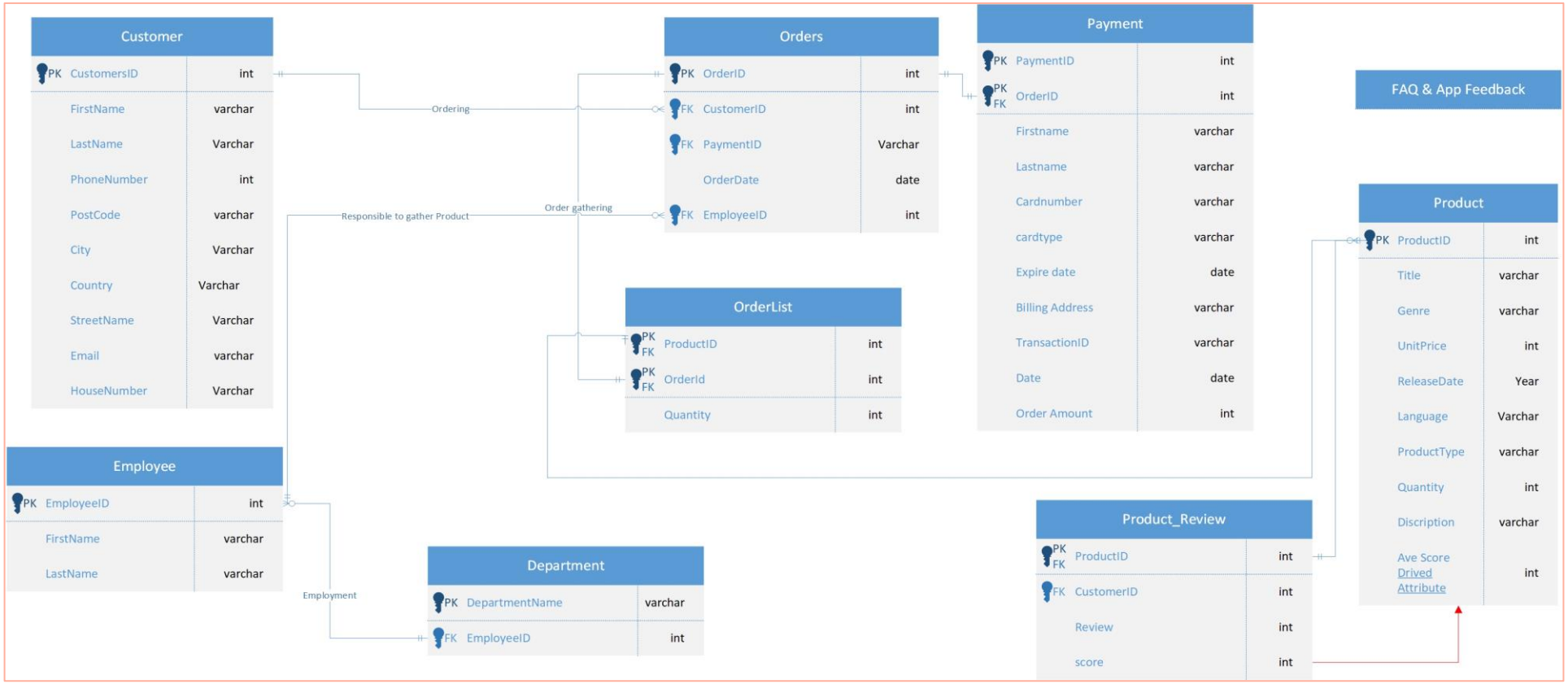
- A widely used convention for representing relationships in Entity Relationship Diagrams (ERDs).
- Employs distinct symbols to visually convey cardinality and modality between entities.
- Enhances clarity and understanding of data structures.

Cardinality-Modality

- Cardinality: Expresses the maximum number of instances of one entity that can be associated with another.
- Modality: Indicates whether participation in a relationship is mandatory or optional.
  - Mandatory: Represented by a solid circle.
  - Optional: Shown as an empty circle.

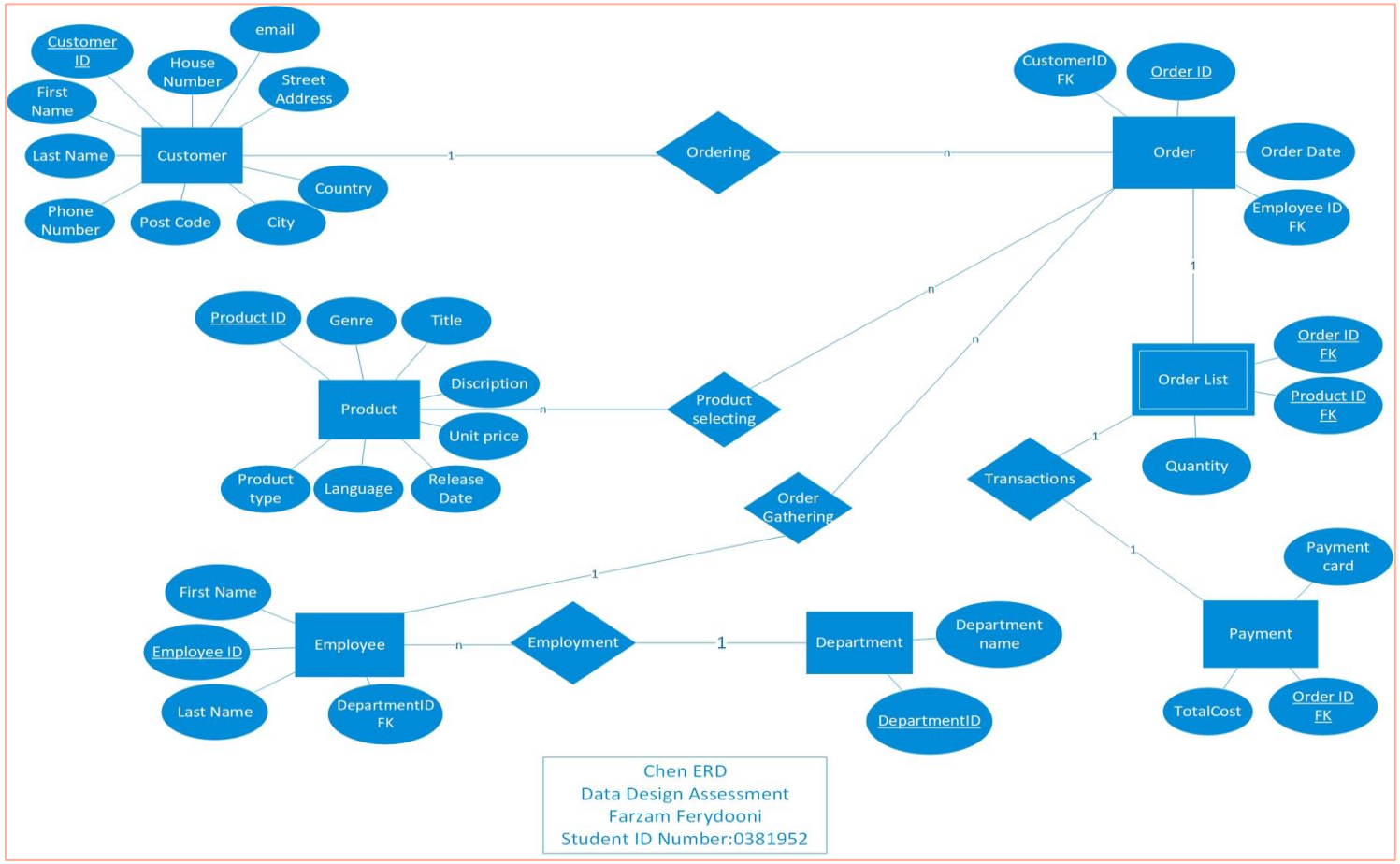
Designing

Crow's Foot Notation
Comprehensive ERD-For Lean Design

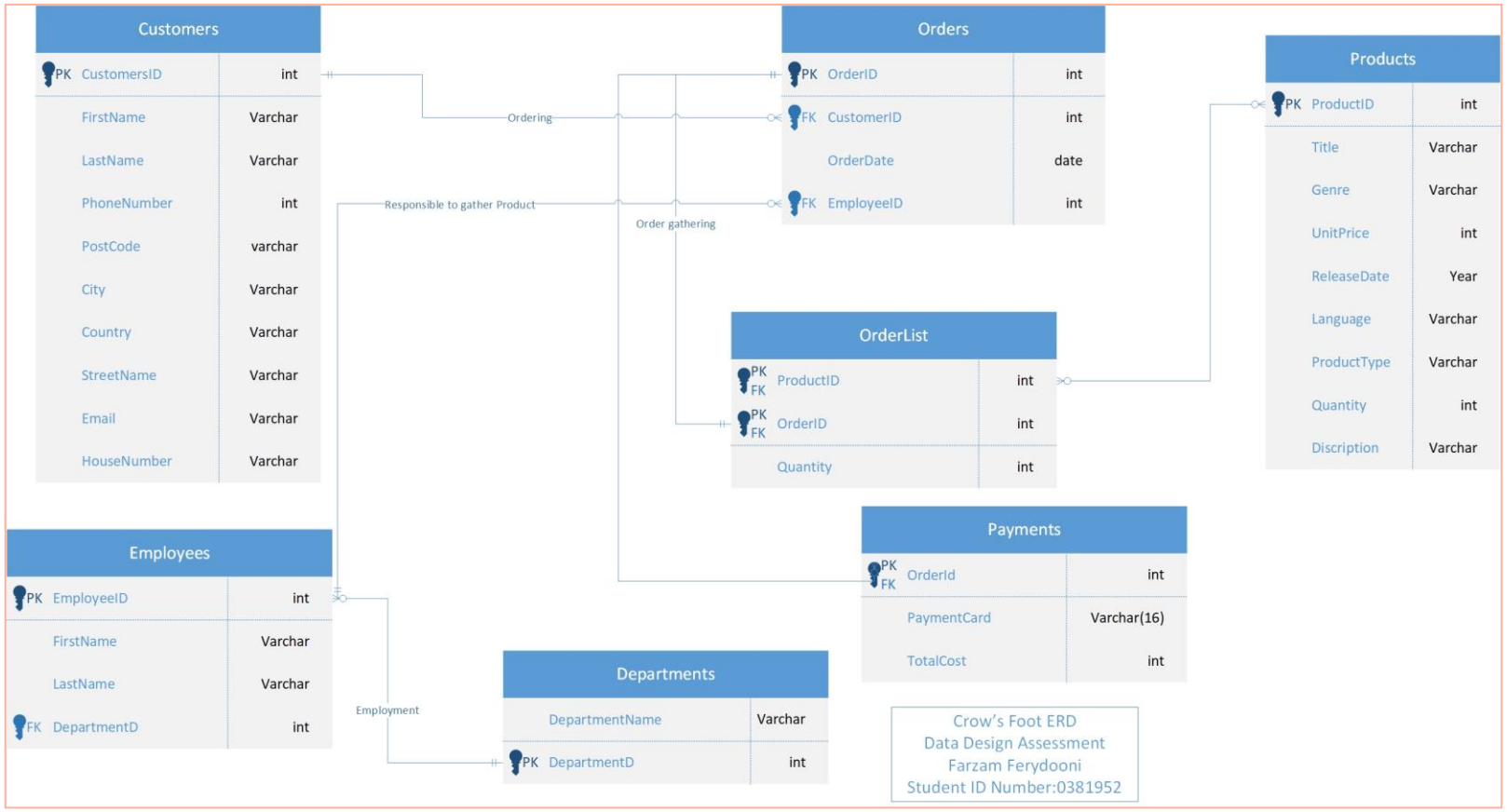


The parts of the assessment question that are desired will be the primary focus of attention since the process of incorporating the features stated on the slide into the framework of the course test is time-consuming. The following slide, which will cover the modules that are necessary for evaluation, is now ready for us to move on to.

Chen Based ERD

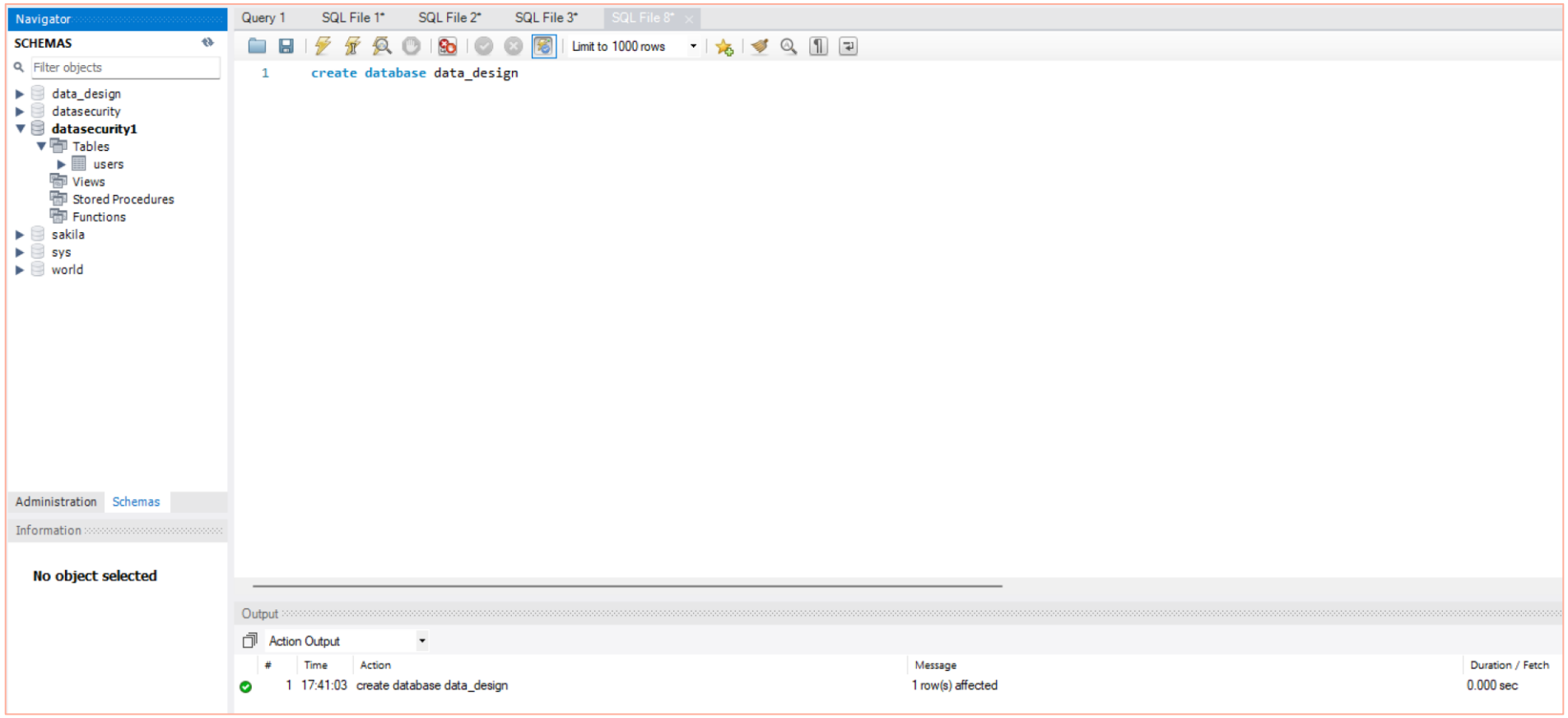


Crow's Foot based ERD



Implementation

Creation Phase
Database



First step is making data base “data_design”

Creation Phase
Customers Table

Query 1 | SQL File 1* | SQL File 2* | SQL File 3* | SQL File 8* | SQL File 9*

Limit to 1000 rows

```sql
CREATE TABLE Customers (
    CustomerID int not null AUTO_INCREMENT,
   FirstName varchar(50),
    LastName varchar(50),
    HomeNumber varchar(10),
    StreetAdd varchar(200),
    City varchar(50),
    Country varchar(50),
    PostCode varchar(20),
    PhoneNumber varchar(20),
    EmailAdd varchar(100),
    primary key (CustomerID)
    );
```

-All Primary Keys in tables are

- NOT NULL
- AUTO_INCREMENT

- For other attributes in tables, Their inputs are controlled within program code

It is urged that we begin the process of creating tables by beginning with a table that has no or a minimal dependent on other tables.

Creation Phase
Department Table

Order Creation | Department Creation | Orderlist creation | SQL File 6* | Emplyee Creation

Limit to 1000 rows

```sql
CREATE TABLE Departments (
    DepartmentID int not null AUTO_INCREMENT,
    DepartmentName varchar(20),
    primary key (DepartmentID)
    );
```

Creation Phase
Products Table

Query 1 | SQL File 1* | SQL File 3* | SQL File 2* | SQL File 8* | SQL File 9*

Limit to 1000 rows

```sql
CREATE TABLE Products (
    ProductID int not null AUTO_INCREMENT,
    Title varchar(100),
    Genre varchar(30),
    Quantity int,
    UnitPrice int,
    ReleaseDate year,
    Lang varchar(30),
    ProductType varchar(20),
    Descriptions varchar(50),
    primary key (ProductID)
    );
```

Output

Action Output

#	Time	Action	Message
1	20:31:40	CREATE TABLE Products (ProductID int not null AUTO_INCREMENT, Title varchar(100), G...	0 row(s) affected

Creation Phase
Employee Table

Order Creation | Department Creation* | Orderlist creation | SQL File 6* | Emplyee Creation

Limit to 1000 rows

```sql
CREATE TABLE Employees (
    EmployeeID int not null AUTO_INCREMENT,
    FirstName varchar(50),
    LastName varchar(50),
    DepartmentID int,
    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID),
    primary key (EmployeeID)
    );
```

Because each employee should connect to one of the departments, the department table is linked to the employee table with a foreign key.

Creation Phase
Orders Table

Order Creation* | Department Creation | Orderlist creation | departments | Order Creation | orders - Table | Payment Creation

Limit to 1000 rows

```sql
CREATE TABLE Orders (
    OrderID int not null AUTO_INCREMENT,
    OrderDate DATE,
    EmployeeID int,
    CustomerID int,
    FOREIGN KEY (CustomerID) REFERENCES customers(CustomerID),
    FOREIGN KEY (EmployeeID) REFERENCES employees(EmployeeID),
    primary key (OrderID)
        );
```

Output

Action Output

#	Time	Action	Message

In Orders table we have two foreign keys, one connects each order to a specific customer and other connect each order to one employee to gathers required products

Creation Phase
OrdersList Table

In this table customers orders should be list, so this table have foreign key to products and orders.

Query 1 | SQL File 1* | SQL File 3* | SQL File 2* | SQL File 8* | SQL File 9*

Limit to 1000 rows

```sql
CREATE TABLE OrderList (
    OrderID int,
    ProductID int,
    Quantity int,
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID),
    primary key (OrderID,ProductID)
        );
```

Output

Action Output

#	Time	Action	Message
1	21:03:30	CREATE TABLE OrderList (OrderID int, ProductID int, Quantity int, FOREIGN KEY (Orderl...	0 row(s) affected

Creation Phase
Payments Table

This table's TotalCost attribute is driven by the OrderList attribute, which gives a list of orders along with their quantities and prices from the Product table.

Order Creation | Department Creation | Orderlist creation | Order Creation | Payment Creation | products | products | Orderlist entry | orders | orders

Limit to 1000 rows

```sql
CREATE TABLE Payments (
    OrderID int,
    PaymentCard varchar(16),
    TotalCost int,
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    primary key (OrderID)
        );
```

Output

Action Output

#	Time	Action	Message
1	21:52:12	CREATE TABLE Payments (OrderID int, PaymentCard varchar(16), TotalCost int, FOREIGN ...	0 row(s) affected

Data Entry

Data Entry Phase
Product

Navigator

SCHEMAS

Filter objects

- data_design
  - Tables
    - customers
    - departments
    - employees
    - orderlist
    - orders
    - products
  - Views
  - Stored Procedures
  - Functions
- datasecurity
- datasecurity1
- sakila
- sys
- world

Administration | Schemas

Information

Connection Details

Name: Local instance MySQL80
Host: localhost
Port: 3306
Login User: root
Current User: root@localhost
SSL cipher: TLS_AES_256_GCM_SHA

Server

Product: MySQL Community Serve
Version: 8.0.35

Connector

Query 1 | SQL File 1* | SQL File 3* | SQL File 2* | SQL File 8* | test | SQL File 10*

Limit to 1000 rows

```sql
INSERT INTO Products (Title, Genre, Quantity, UnitPrice, ReleaseDate, Lang, ProductType, Descriptions)
VALUES
  -- Movies
  ('The Lord of the Rings: The Fellowship of the Ring', 'Fantasy', 5, 4, 2001, 'English', 'Movie', 'An epic fantasy adventure'),
  ('The Shawshank Redemption', 'Drama', 3, 3, 1994, 'English', 'Movie', 'A critically acclaimed drama about hope and redemption'),
  ('The Dark Knight', 'Action', 8, 5, 2008, 'English', 'Movie', 'A gritty and intense superhero crime thriller'),
  ('The Godfather', 'Crime Drama', 2, 4, 1972, 'English', 'Movie', 'A classic gangster film about family, power, and betrayal'),
  ('The Matrix', 'Sci-Fi', 7, 3, 1999, 'English', 'Movie', 'A mind-bending action film that explores the nature of reality'),
  -- Series
  ('Friends', 'Comedy', 10, 2, 1994, 'English', 'TV Series', 'A sitcom about six friends in New York City'),
  ('The Office', 'Mockumentary Comedy', 6, 3, 2005, 'English', 'TV Series', 'A mockumentary about office life in Scranton, Pennsylvania'),
  ('Breaking Bad', 'Crime Drama', 4, 4, 2008, 'English', 'TV Series', 'A high-stakes drama about a chemistry teacher turned drug kingpin'),
  ('Game of Thrones', 'Fantasy', 8, 5, 2011, 'English', 'TV Series', 'An epic fantasy drama about power struggles in a medieval world'),
  ('The Wire', 'Crime Drama', 2, 4, 2002, 'English', 'TV Series', 'A gritty and realistic portrayal of the drug trade in Baltimore'),
  -- Music
  ('Abbey Road', 'Rock', 15, 2, 1969, 'English', 'Album', 'The Beatles final studio album'),
  ('Thriller', 'Pop', 12, 3, 1982, 'English', 'Album', 'Michael Jackson best-selling album'),
  ('The Dark Side of the Moon', 'Progressive Rock', 10, 4, 1973, 'English', 'Album', 'Pink Floyd landmark album'),
  ('Kind of Blue', 'Jazz', 8, 3, 1959, 'English', 'Album', 'Miles Davis masterpiece'),
  ('Nevermind', 'Grunge', 6, 4, 1991, 'English', 'Album', 'Nirvana iconic album that changed the course of rock music');
```

Output

Action Output

#	Time	Action	Message
1	18:25:18	INSERT INTO Products (Title, Genre, Quantity, UnitPrice, ReleaseDate, Lang, ProductType, Descript...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0

Data Entry Phase
Customers

As mentioned earlier, when a row in a table is given a value, the primary key is automatically assigned.

Customers creation | SQL File 3* | SQL File 2* | SQL File 8* | test | SQL File 11*

Limit to 1000 rows

```sql
INSERT INTO Customers (FirstName, LastName, HomeNumber, StreetAdd, City, Country, PostCode, PhoneNumber, EmailAdd)
VALUES
  ('John', 'Doe', '25', '123 Main St', 'Anytown', 'UK', 'AB12 3CD', '07123456789', 'johndoe@example.com'),
  ('Jane', 'Smith', '42', '456 Elm St', 'New City', 'USA', '12345', '123-456-7890', 'janesmith@example.com'),
  ('Bob', 'Johnson', '11A', '789 Oak St', 'Springfield', 'UK', 'DE5 4FG', '01234567890', 'bobjohnson@example.com'),
  ('Emily', 'Davis', '3B', '555 Maple St', 'Metropolis', 'USA', '98765', '555-555-5555', 'emilydavis@example.com'),
  ('Michael', 'Brown', '22', '888 Pine St', 'Riverdale', 'UK', 'M1 1AA', '09876543210', 'michaelbrown@example.com'),
  ('Sarah', 'Miller', '6', '999 Cedar St', 'Sunnyville', 'USA', '43210', '666-666-6666', 'sarahmiller@example.com'),
  ('David', 'Williams', '1C', '111 Birch St', 'Greentown', 'UK', 'N1 2BB', '04445556667', 'davidwilliams@example.com'),
  ('Lisa', 'Jones', '54', '222 Willow St', 'Hill Valley', 'USA', '32109', '777-777-7777', 'lisajones@example.com'),
  ('James', 'White', '99', '333 Oakwood St', 'Bayside', 'UK', 'SW1A 1AA', '08889990001', 'jameswhite@example.com'),
  ('Mary', 'Taylor', '101', '444 Elmwood St', 'Star City', 'USA', '09876', '888-888-8888', 'marytaylor@example.com');
```

Output

Action Output

#	Time	Action	Message
1	18:45:52	INSERT INTO Customers (FirstName, LastName, HomeNumber, StreetAdd, City, Country, PostCode, ...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0

Data Entry Phase
Employee

Customers creation | SQL File 3* | SQL File 2* | Emplyee Creation | test | Customers Data entry | Emplyee entry

Limit to 1000 rows

```sql
INSERT INTO Employees (FirstName, LastName)
VALUES
  ('Alice', 'Smith'),
  ('Bob', 'Johnson'),
  ('Charlie', 'Davis'),
  ('Emily', 'Brown'),
  ('David', 'Miller');
```

Output

Action Output

#	Time	Action	Message
1	18:45:52	INSERT INTO Customers (FirstName, LastName, HomeNumber, StreetAdd, City, Country, PostCode, ...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0
2	19:04:23	INSERT INTO Employees (FirstName, LastName) VALUES ('Alice', 'Smith'), ('Bob', 'Johnson'), ('Ch...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0

Data Entry Phase
Departments

```sql
INSERT INTO departments (DepartmentName)
VALUES
  ('Deliveries'),
  ('Accounting'),
  ('Human_Resources');
```

Data Entry Phase
Orders

In this section, in order to make each row clear, the given value has been related to the Employee table with explanations.

Order Creation | Department Creation | Orderlist creation | Order Creation | orders - Table | Payment Creation | SQL File 14* | products | Order entry | empl

a new view in the active schema in the connected server | Limit to 1000 rows

```sql
INSERT INTO Orders (OrderDate, EmployeeID, CustomerID)
VALUES
  ('2023-12-15', 6, 1),  -- EmployeeID 6 (Alice Smith)
  ('2023-11-01', 7, 3),  -- EmployeeID 7 (Bob Johnson)
  ('2023-12-05', 8, 8),  -- EmployeeID 3 (Charlie Davis)
  ('2023-12-15', 6, 7),  -- EmployeeID 6 (Alice Smith)
  ('2023-10-26', 6, 3),  -- EmployeeID 6 (Alice Smith)
  ('2023-11-10', 8, 5);  -- EmployeeID 3 (Charlie Davis)
```

Output

Action Output

#	Time	Action	Message
1	19:59:26	INSERT INTO Orders (OrderDate, EmployeeID, CustomerID) VALUES ('2023-12-15', 6, 1), -- Employe...	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0

Data Entry Phase

Order Creation | Department Creation | Orderlist creation | Order Creation | Payment Creation | products | products | Orderlist entry | orders | orders

Limit to 1000 rows

```sql
INSERT INTO OrderList (OrderID, ProductID, Quantity)
VALUES
  (15, 19, 2),  (15, 18, 1),  (17, 20, 3),  (18, 26, 1),  (19, 26, 1),  (20, 20, 2),
  (19, 18, 2),  (18, 19, 2),  (15, 28, 1),  (20, 30, 1);
```

Data Entry Phase Payments-Part1

As mentioned before, the Payments table is calculated from the combination of OrderList and Products. Therefore, in this section, the cost of each order item is calculated using a JOIN operation and then multiplied. Then, the total order amount is calculated using the programmer's instructions (as shown in the next slide).

```sql
use data_design;
 SELECT
    Orderlist.ProductID,
    (Products.unitprice * Orderlist.Quantity) AS TotalItemsOrdered
FROM Products
    JOIN Orderlist
        ON Products.ProductID = Orderlist.ProductID
ORDER BY Orderlist.ProductID;
```

Result Grid

ProductID	TotalItemsOrdered
18	5
18	10
19	8
19	8
20	9
20	6
26	2
26	2
28	4
30	4

Output

#	Time	Action	Message
1	19:37:21	use data_design	0 row(s) affected
2	19:37:21	SELECT Orderlist.ProductID, (Products.unitprice * Orderlist.Quantity) AS TotalItemsOrdered FRO...	10 row(s) returned

Data Entry Phase
Payments-Part2

cts | Orderlist entry | orders | payments | orders | orderlist | SQL File 17* | orderlist | orderlist | customers | users | users - Table | users | p

Limit to 1000 rows

```sql
INSERT INTO Payments (OrderID, PaymentCard, TotalCost)
VALUES
  (15, '5694313011065898', 17),
  (17, '5935438191715898', 9),
  (18, '4512372615018563', 10),
  (19, '7331556147874563', 12),
  (20, '6508706616014563', 10);
```

Output

Action Output

#	Time	Action	Message
1	19:57:02	INSERT INTO Payments (OrderID, PaymentCard, TotalCost) VALUES (15, '5694313011065898', 17), ...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0

Current State of Tables

Tables State Customers

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

CustomerID	FirstName	LastName	HomeNumber	StreetAdd	City	Country	PostCode	PhoneNumber	EmailAdd
1	John	Doe	25	123 Main St	Anytown	UK	AB12 3CD	07123456789	johndoe@example.com
2	Jane	Smith	42	456 Elm St	New City	USA	12345	123-456-7890	janesmith@example.com
3	Bob	Johnson	11A	789 Oak St	Bayside	UK	DE5 4FG	01234567890	bobjohnson@example.com
4	Emily	Davis	3B	555 Maple St	Metropolis	USA	98765	555-555-5555	emilydavis@example.com
5	Michael	Brown	22	888 Pine St	Riverdale	UK	M1 1AA	09876543210	michaelbrown@example.com
6	Sarah	Miller	6	999 Cedar St	Sunnyville	USA	43210	666-666-6666	sarahmiller@example.com
7	David	Williams	1C	111 Birch St	Greentown	UK	N1 2BB	04445556667	davidwilliams@example.com
8	Lisa	Jones	54	222 Willow St	Hill Valley	USA	32109	777-777-7777	lisajones@example.com
9	James	White	99	333 Oakwood St	Bayside	UK	SW1A 1AA	08889990001	jameswhite@example.com
10	Mary	Taylor	101	444 Elmwood St	Star City	USA	09876	888-888-8888	marytaylor@example.com
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

customers 1

Tables State
Departments-Employees-Orders

Result Grid | Filter Rows: | Edit:

DepartmentID	DepartmentName
1	Deliveries
2	Accounting
3	Human_Resources
NULL	NULL

departments 2

Result Grid | Filter Rows: | Edit:

EmployeeID	FirstName	LastName	DepartmentID
6	Alice	Smith	1
7	Bob	Johnson	3
8	Charlie	Davis	1
9	Emily	Brown	2
10	David	Miller	3
NULL	NULL	NULL	NULL

employees 3

Result Grid | Filter Rows: | Edit:

OrderID	OrderDate	EmployeeID	CustomerID
15	2023-12-15	6	1
16	2023-11-01	7	3
17	2023-12-05	8	8
18	2023-12-15	6	7
19	2023-10-26	6	3
20	2023-11-10	8	5
NULL	NULL	NULL	NULL

orders 4

Tables State
OrderList-Payments

Result Grid | Filter Rows: | Edit:

OrderID	ProductID	Quantity
15	18	1
15	19	2
15	28	1
17	20	3
18	19	2
18	26	1
19	18	2
19	26	1
20	20	2
20	30	1
NULL	NULL	NULL

orderlist 5

Result Grid | Filter Rows: | Edit:

OrderID	PaymentCard	TotalCost
15	5694313011065898	17
17	5935438191715898	9
18	4512372615018563	10
19	7331556147874563	12
20	6508706616014563	10
NULL	NULL	NULL

payments 6

Tables State
Products

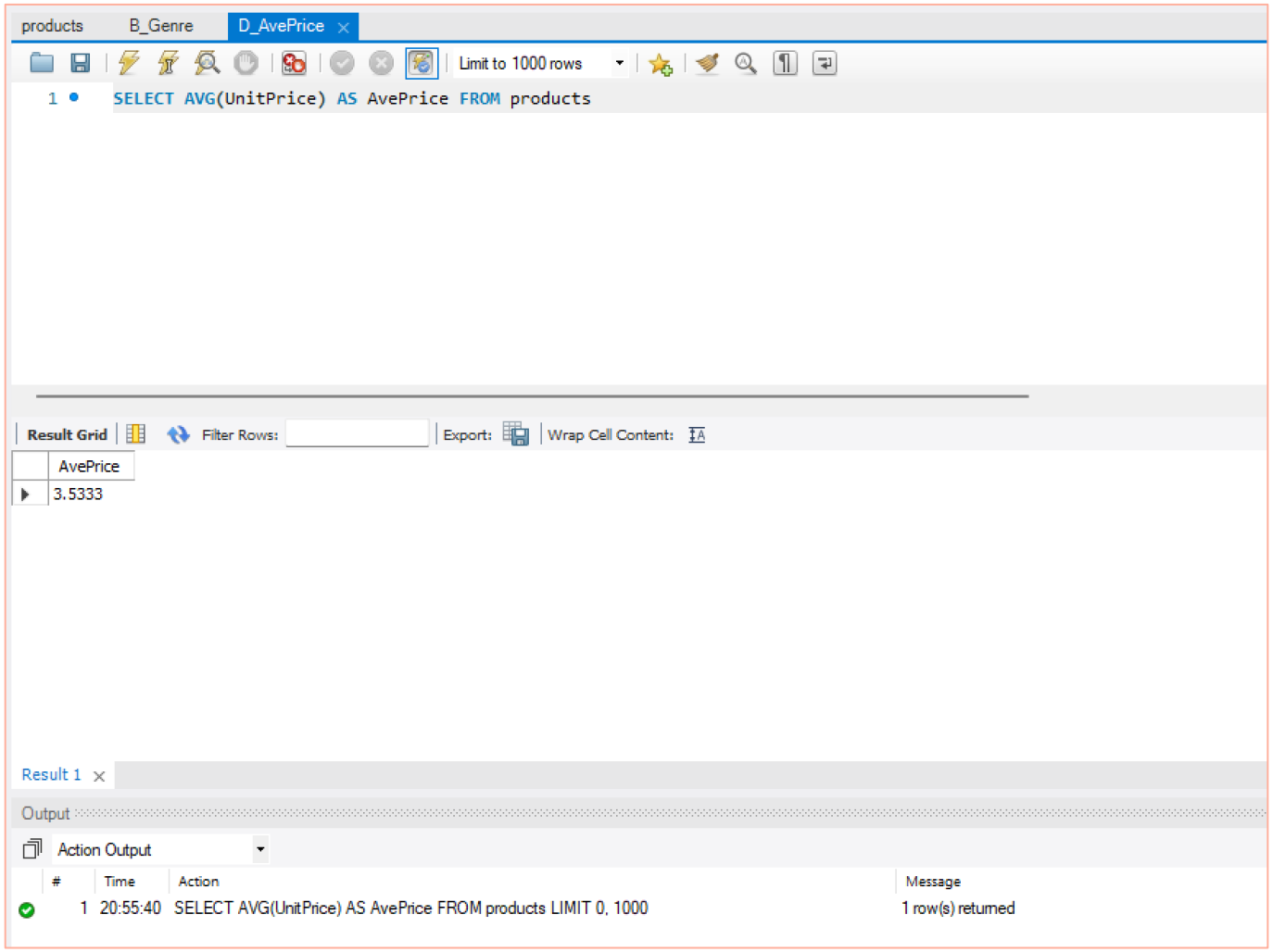
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ProductID	Title	Genre	Quantity	UnitPrice	ReleaseDate	Lang	ProductType	Descriptions
16	The Lord of the Rings: The Fellowship of the Ring	Fantasy	5	4	2001	English	Movie	An epic fantasy adventure
17	The Shawshank Redemption	Drama	3	3	1994	English	Movie	A critically acclaimed drama about hope and red...
18	The Dark Knight	Action	8	5	2008	English	Book	A gritty and intense superhero crime thriller
19	The Godfather	Crime Drama	2	4	1972	English	Movie	A classic gangster film about family, power, and...
20	The Matrix	Sci-Fi	7	3	1999	English	Movie	A mind-bending action film that explores the nat...
21	Friends	Comedy	10	2	1994	English	TV Series	A sitcom about six friends in New York City
22	The Office	Mockumentary Comedy	6	3	2005	English	TV Series	A mockumentary about office life in Scranton, P...
23	Breaking Bad	Crime Drama	4	4	2008	English	TV Series	A high-stakes drama about a chemistry teacher ...
24	Game of Thrones	Fantasy	8	5	2011	English	TV Series	An epic fantasy drama about power struggles in...
25	The Wire	Crime Drama	2	4	2002	English	TV Series	A gritty and realistic portrayal of the drug trade...
26	Abbey Road	Rock	15	2	1969	English	Album	The Beatles final studio album
27	Thriller	Pop	12	3	1982	English	Album	Michael Jackson best-selling album
28	The Dark Side of the Moon	Progressive Rock	10	4	1973	English	Album	Pink Floyd landmark album
29	Kind of Blue	Jazz	8	3	1959	English	Album	Miles Davis masterpiece
30	Nevermind	Grunge	6	4	1991	English	Album	Nirvana iconic album that changed the course o...
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

products 7

Assessment Tasks

A-Extract all the customers from a specific city



B-Search for a product of a specific genre

products | B_Genre

Limit to 1000 rows

```
1 •  SELECT * FROM products WHERE Genre='Crime Drama'
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ProductID	Title	Genre	Quantity	UnitPrice	ReleaseDate	Lang	ProductType	Descriptions
19	The Godfather	Crime Drama	2	4	1972	English	Movie	A classic gangster film about family, power, and...
23	Breaking Bad	Crime Drama	4	4	2008	English	TV Series	A high-stakes drama about a chemistry teacher ...
25	The Wire	Crime Drama	2	4	2002	English	TV Series	A gritty and realistic portrayal of the drug trade...
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

products 1

Output

Action Output

#	Time	Action	Message
1	20:42:30	SELECT count(FirstName) AS CountOfrepeatedCity FROM Customers WHERE City='Bayside' LIMIT 0,...	1 row(s) returned
2	20:44:52	SELECT * FROM data_design.products LIMIT 0, 1000	15 row(s) returned
3	20:50:27	SELECT * FROM products WHERE Genre='Crime Drama' LIMIT 0, 1000	3 row(s) returned

C-Count how many customers are from a specific city

customers | a-select_city | SQL File 4*

Limit to 1000 rows

```
1 • SELECT count(FirstName) AS CountOfrepeatedCity FROM Customers WHERE City='Bayside'
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CountOfrepeatedCity
2

Result 2

Output

Action Output

#	Time	Action	Message
1	20:42:30	SELECT count(FirstName) AS CountOfrepeatedCity FROM Customers WHERE City='Bayside' LIMIT 0,...	1 row(s) returned

D-Calculate the average of the unit price

products | B_Genre | D_AvePrice

Limit to 1000 rows

```
1 •  SELECT AVG(UnitPrice) AS AvePrice FROM products
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AvePrice
3.5333

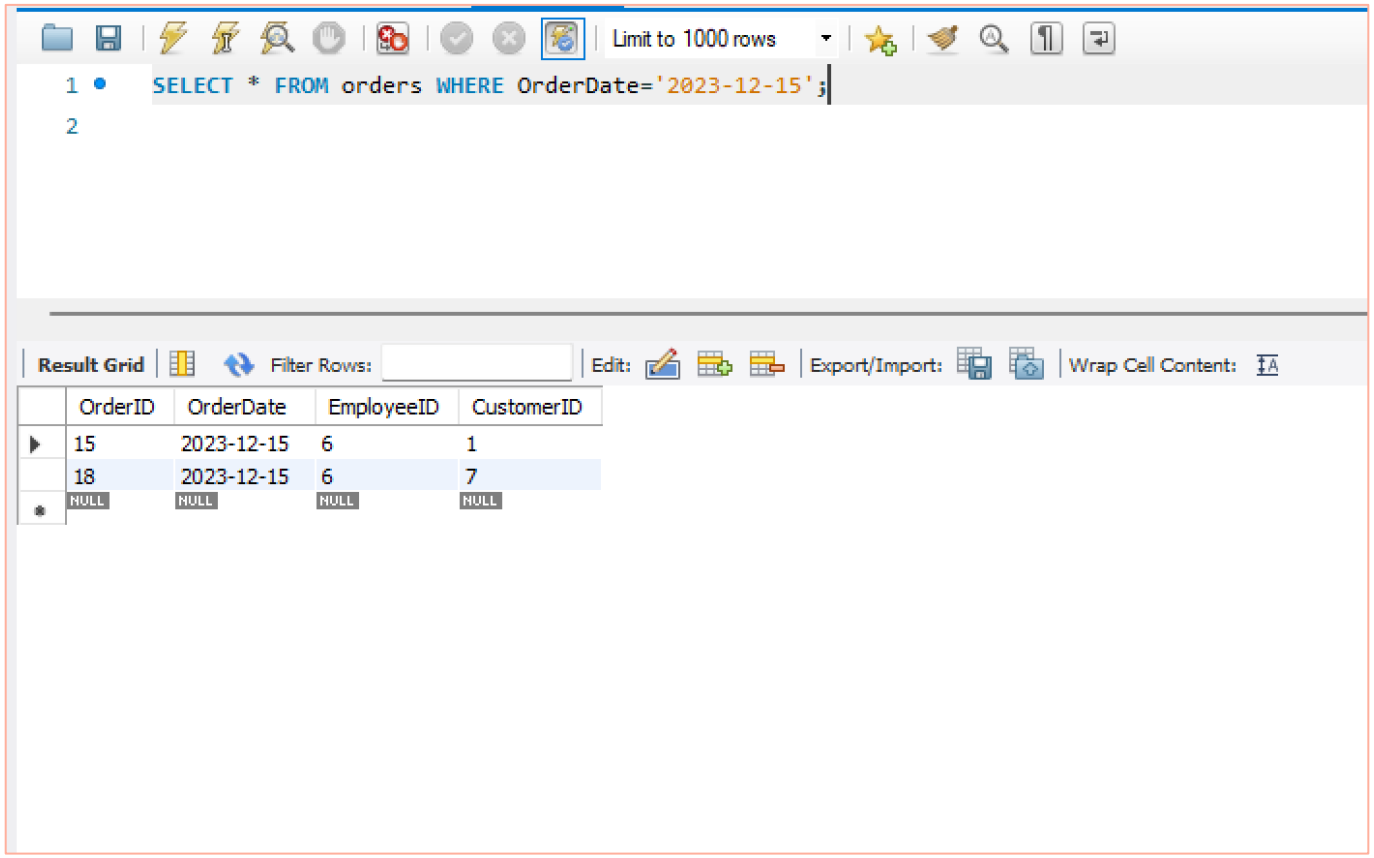
Result 1

Output

Action Output

#	Time	Action	Message
1	20:55:40	SELECT AVG(UnitPrice) AS AvePrice FROM products LIMIT 0, 1000	1 row(s) returned

E-Extract all current orders.



Assumption: 2023-12-15 is current date

E-Extract all current orders-Complit Tables.

products | orders | SQL File 8* | E-Current | SQL File 11* | SQL File 12*

Limit to 1000 rows

```sql
SELECT orders.OrderID,orders.EmployeeID,orders.OrderDate,orderlist.Quantity,products.Title,orders.OrderDate
 FROM orders JOIN orderlist ON orderlist.OrderID=orders.OrderID JOIN products on products.ProductID=orderlist.ProductID WHERE orders.OrderDate='2023-12-15';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

OrderID	EmployeeID	OrderDate	Quantity	Title	OrderDate
15	6	2023-12-15	1	The Dark Knight	2023-12-15
15	6	2023-12-15	2	The Godfather	2023-12-15
15	6	2023-12-15	1	The Dark Side of the Moon	2023-12-15
18	6	2023-12-15	2	The Godfather	2023-12-15
18	6	2023-12-15	1	Abbey Road	2023-12-15

Assumption: 2023-12-15 is current date

F-Extract all orders for books that have the keyword “the” in their description

products orders F_The

Limit to 1000 rows

```
1  SELECT OrderList.OrderID, Products.Title
2  FROM Products
3  INNER JOIN OrderList ON OrderList.ProductID = Products.ProductID
4  WHERE Products.Title LIKE 'The%' AND Products.ProductType='Book' ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

OrderID	Title
15	The Dark Knight
19	The Dark Knight

Result 7

Output

Action Output

#	Time	Action	Message
1	22:56:30	SELECT OrderList.OrderID, Products.Title FROM Products INNER JOIN OrderList ON OrderList.Produ...	2 row(s) returned

G-Extract all payments with credit cards for music records

products | orders | SQL File 8* | E-Current | SQL File 12* | employees | payments | G-PaymentCard

Limit to 1000 rows

```
1 • SELECT orders.OrderID,orders.EmployeeID,orders.OrderDate,orderlist.Quantity,products.Title,orders.OrderDate,payments.PaymentCard  FROM orders JOIN orderlist ON
orderlist.OrderID=orders.OrderID JOIN products on products.ProductID=orderlist.ProductID JOIN payments ON orderlist.OrderID=payments.OrderID WHERE products.
ProductType='Album' and LENGTH (payments.PaymentCard) = 16 AND payments.PaymentCard REGEXP '^[0-9]+$';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

OrderID	EmployeeID	OrderDate	Quantity	Title	OrderDate	PaymentCard
18	6	2023-12-15	1	Abbey Road	2023-12-15	4512372615018563
19	6	2023-10-26	1	Abbey Road	2023-10-26	7331556147874563
15	6	2023-12-15	1	The Dark Side of the Moon	2023-12-15	5694313011065898
20	8	2023-11-10	1	Nevermind	2023-11-10	6508706616014563

Result Grid

Form Editor

H-Count how many employees handle music records

SQL File 8* | E-Current | C_repeatcity | SQL File 12* | J-CountOrders | employees | I-CountOfEmployeesWithS | H-CountEmployee

Limit to 1000 rows

```
1 •  SELECT count(DISTINCT orders.EmployeeID) as SUMER FROM orders JOIN orderlist ON orderlist.OrderID=orders.
     OrderID JOIN products on products.ProductID=orderlist.ProductID WHERE products.ProductType='Album' ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

SUMER
2

Result Grid

Form

I-Count the employees with the first name starting with the letter S

products | orders | SQL File 8* | E-Current | C_repeatcity | SQL File 12* | J-CountOrders | employees | I-CountOfEmployeesWithS

Limit to 1000 rows

```
1 SELECT count(EmployeeID) AS CountOfEmployeesWithS FROM employees WHERE FirstName like "S%" OR FirstName like "s%"
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	CountOfEmployeesWithS
▶	0

J-Count how many orders are in the system

products | orders | SQL File 8* | E-Current | C_repeatcity | SQL File 12* | J-CountOrders

Limit to 1000 rows

```
1 •  SELECT count(OrderID) AS CountOfOrders FROM orders
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	CountOfOrders
▶	6

Use Cases

Use Cases-1
Get details of a customer order

```sql
SELECT customers.FirstName, customers.LastName, orders.OrderDate, payments.TotalCost
FROM customers
JOIN orders ON orders.CustomerID = Customers.CustomerID
JOIN payments ON payments.OrderID = orders.OrderID
WHERE orders.OrderID = 15;
```

Result Grid

FirstName	LastName	OrderDate	TotalCost
John	Doe	2023-12-15	17

Result 1

Output

Action Output

#	Time	Action	Message
1	17:31:10	SELECT customers.FirstName, customers.LastName, orders.OrderDate, payments.TotalCost FROM c...	1 row(s) returned

Use Cases-2
Get total quantity sold of each product:

Limit to 1000 rows

```
SELECT products.ProductID, products.Title, SUM(orderlist.Quantity) AS TotalSold
FROM orderlist
JOIN products ON products.ProductID = orderlist.ProductID
GROUP BY orderlist.ProductID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ProductID	Title	TotalSold
18	The Dark Knight	3
19	The Godfather	4
28	The Dark Side of the Moon	1
20	The Matrix	5
26	Abbey Road	2
30	Nevermind	1

Result 1

Output

Action Output

#	Time	Action	Message
1	17:47:03	SELECT products.ProductID, products.Title, SUM(orderlist.Quantity) AS TotalSold FROM orderlist JO...	6 row(s) returned

Use Cases-3
Get customers who have made at least 2 purchases in the last 3 months:

- The DATE_SUB() function is used to subtract a time/date interval from a given date and retrieve the resulting date(MySQL DATE_SUB() Function, no date).
- The CURDATE() function returns the current date in numeric or string format(MySQL CURDATE() Function, no date).
- Selected INTERVAL is 3 month to show query functionality.

Limit to 1000 rows

```
FROM customers
JOIN orders ON customers.CustomerID = orders.CustomerID
WHERE orders.OrderDate >= DATE_SUB(CURDATE(), INTERVAL 3 MONTH)
GROUP BY customers.CustomerID
HAVING COUNT(orders.OrderID) >= 2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CustomerID	FirstName	LastName	OrderCount
3	Bob	Johnson	2

Result 7

Output

Action Output

#	Time	Action	Message
1	19:43:43	SELECT customers.CustomerID, FirstName, LastName, COUNT(OrderID) AS OrderCount FROM custo...	1 row(s) returned

Use Cases-4
Identify customers who haven't made a purchase in the last 2 month

```sql
SELECT customers.CustomerID, customers.FirstName, customers.LastName, MAX(orders.OrderDate) AS LastOrderDate
FROM customers
LEFT JOIN orders ON customers.CustomerID = orders.CustomerID
GROUP BY customers.CustomerID
HAVING MAX(orders.OrderDate) < DATE_SUB(CURDATE(), INTERVAL 2 MONTH);
```

Result Grid

CustomerID	FirstName	LastName	LastOrderDate
3	Bob	Johnson	2023-11-01
5	Michael	Brown	2023-11-10

Result 3

Output

Action Output

#	Time	Action	Message
1	19:58:41	SELECT customers.CustomerID, customers.FirstName, customers.LastName, MAX(orders.OrderDate) ...	2 row(s) returned

MAX() is a function with find maximum in a list.

Use Cases-5
get the name and ID of the department that has the most employees

```
SELECT DepartmentName, DepartmentID FROM departments
WHERE DepartmentID = (SELECT DepartmentID FROM employees GROUP BY DepartmentID ORDER BY COUNT(*) DESC LIMIT 1)
```

Result Grid

DepartmentName	DepartmentID
Deliveries	1
NULL	NULL

departments 2

Output

Action Output

#	Time	Action	Message
1	20:21:19	SELECT DepartmentName, DepartmentID FROM departments WHERE DepartmentID = (SELECT De...	1 row(s) returned

Finally

Conclusion-1

- In summary, this project involved designing a database to support an online shop selling books, movies, and music records worldwide. Several key considerations were taken into account during the design process, including performance, scalability, security, and maintainability. Lean principles were applied to streamline the design and ensure it delivers maximum value to customers.

Conclusion-2

- Prior to executing example use cases to get significant insights, test data was inputted to populate the tables. In summary, the well-considered design allows the database to effectively meet the operational and analytical requirements of the online shop.

Reflection

- This database design showed me how to use classroom principles of requirements analysis, data modelling, normalisation, and SQL queries. I learned how to design methodically around end-user needs. Lean concepts also helped me maximise value rather than merely finish features.
- In the future, I would focus more on security and access control than during this deployment. Considering flexibility and extensibility beforehand has positioned the database for future expansion as the online shop's offers grow. Use case testing yielded valuable ideas, but more validation is needed. This project greatly improved my database design skills through hands-on learning.

References

- Lui M [Photograph] 2009 Available at: https://www.timeout.com/sydney/shopping/title-music-film-books-surry-hills.
- Silberschatz, Abraham, et al. Database System Concepts. New York, Ny, Mcgraw-Hill Education, 2020, pp. 1–25.
- Graupp, P. (2022) What Are Lean Operations? Everything You Need to Know, TWI Institute. Available at: https://www.twi-institute.com/lean-operations/ (Accessed: 10 January 2024).
- MySQL DATE_SUB() Function (no date). Available at: https://www.w3schools.com/sql/func_mysql_date_sub.asp (Accessed: 23 January 2024).
- MySQL CURDATE() Function (no date). Available at: https://www.w3schools.com/sql/func_mysql_curdate.asp (Accessed: 23 January 2024).