BASIC TASKS

1. Define the following terms:

a. Candidate key: A collection of attributes that uniquely identifies each record in a database table.

b. Composite key: a key that consists of two or more attributes that uniquely identifies each row in a relationship. Composite keys can be used when a single attribute cannot uniquely identify a row.

c. Foreign key: A foreign key is used to refer to the primary key of another table. It establishes a link between the two tables, ensuring data consistency and referential integrity.

d. Functional dependency: Functional dependencies describe that the value of one property or group of properties determines the value of another property or set of properties.

2. Identify and define the three integrity rules/constraints in the relational model

Entity Integrity：

Entity integrity requires that each tuple (row) in a relationship be unique, usually through a primary key. The value of the primary key must be unique and cannot be NULL, which ensures the unique identification of each row of data.

Referential Integrity：

It involves associations between two or more tables. If the foreign key of a table (child table) references the primary key of another table (parent table), you must ensure that the value of the foreign key either matches one of the primary keys of the parent table or is NULL. This prevents referencing non-existent records.

Domain Integrity：

Domain integrity is a constraint on the value of a column, which specifies the data type and possible range of values for the column. This completeness can be enforced by setting CHECK constraints .

3. For a given multiple relational table, relational integrity rules play an important role in ensuring data accuracy and consistency.

a.Table 1:

Table I violates the entity integrity rules. Entity integrity requires that the primary key cannot be null or partially null, and the missing data for the column name genre in the row with the primary key filmNo of 008 could result in the record not being accurately identified and processed under certain operations.

b.Table 2:

Table II violates both domain integrity rules and entity integrity rules. Domain integrity requires that the data in a column must conform to the requirements of a specific data type, where the data type of the column name Quantity should be numeric, but there is a non-compliant value such as ‘ABC’; at the same time, for the combination of the primary key of Supplier and PartNo, the data given is Meanwhile, for the combination of Supplier and PartNo as primary key, the given data ‘1234,67A,67B,Cd12,0242,AB/1’ violates the entity integrity rules. Entity integrity requires that the value of the primary key cannot be null and uniquely identifies a record. In the given data, there is a mixed combination of letters and numbers and even special characters, which makes it difficult to ensure the uniqueness of the data form, and cannot accurately and uniquely identify each record, and does not comply with the requirements of the domain integrity, because the data form is not uniform, and it is not possible to determine its explicit data type.

c.Table 3:

Table 3 violates the rules of entity integrity. Entity integrity requires that the primary key not be null or partially null, and the missing data for another primary key named Artist in a row with a primary key of Song affects the uniqueness and accuracy of that record.

MEDIUM TASKS

4. Identify the anomalies that may result from the following operations on the table (hint: INSERT, UPDATE, DELETE anomalies)

a. If the ProjectManager field (M. Uhura) does not have a corresponding record in EmployeeNo and EmployeeName, then this insert will violate the referential integrity rule.
b. If there are any employee records associated with the "PrC10" project (if there is a foreign key referencing the project) the deletion operation will violate referential integrity rules, unless the table structure allows cascading deletion or all associated records have been processed.
c. It may violates referential integrity.
d. This table meets the basic requirements of 1NF, which means that each column is atomic and each row of data has a unique identifier (primary key). However, if there is redundant or duplicated data, such as ProjectTitle and ProjectCode being used together as primary keys, it may indicate incomplete normalization.
e. In order to achieve 2NF, all non primary attributes in the relationship must be completely dependent on the entire composite primary key. In the current table structure, there may be partial dependencies, such as DepartmentName depending on DepartmentNo, but ProjectTitle may only depend on ProjectCode, which violates the requirement of 2NF.
f. To achieve 3NF, in addition to satisfying 2NF, it is also necessary to ensure that no dependencies are passed in the relationship. In the current table structure, if there are any indirect dependencies (such as DepartmentName depending on ProjectCode through DepartmentNo), additional relational tables need to be created for decomposition to eliminate these transitive dependencies.

🍞5.

a. Identify exceptions

1. Insertion exceptions

If you want to insert a new customer information, but the customer does not have an order yet, it will lead to an insertion exception because the information related to the order (such as order number, product, quantity, etc.) is mixed with the customer information in the table, and the customer information can not be inserted separately.

2. Update exceptions

For example, if a customer's address changes, because the same customer may have multiple order records, you need to update the address information in multiple records, which is prone to inconsistent updating, leading to update exceptions.

3. Deletion Exception

If a certain order record of a customer is deleted, the information related to that customer may be deleted by mistake, because the customer information and order information are not stored separately, leading to deletion anomalies.

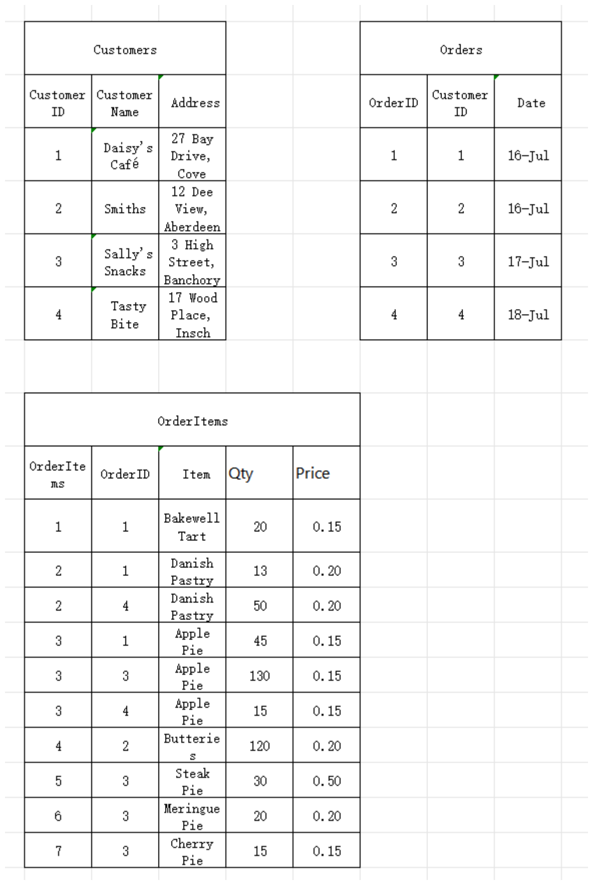
b. Derivation 1NF

1. Analysis

The first paradigm requires that each field is atomic and not subdividable. Observing the table, the data basically meets the atomicity requirement, but duplicate groups (e.g., different items and their quantities corresponding to the same order number) need to be split.

2. Steps

- Create three tables: customer table (Customers), order table (Orders) and order items table (OrderItems).
- Customers table:
  - Columns: CustomerID, which can be incremented as a primary key, CustomerName, and Address.
- Orders table (Orders):
  - Columns: order number (OrderID, can be incremented as the primary key), customer number (CustomerID, as a foreign key associated with the customer table), the date of the order (Date), the total price of the order (TotalCost)
- Order items table (OrderItems):
  - Columns: OrderItemID (can be incremented as the primary key), OrderID (as a foreign key associated with the Order table), Item, Quantity, Price.



Customers		
Customer ID	Customer Name	Address
1	Daisy's Café	27 Bay Drive, Cove
2	Smiths	12 Dee View, Aberdeen
3	Sally's Snacks	3 High Street, Banchory
4	Tasty Bite	17 Wood Place, Insch

Orders		
OrderID	Customer ID	Date
1	1	16-Jul
2	2	16-Jul
3	3	17-Jul
4	4	18-Jul

OrderItems				
OrderItems	OrderID	Item	Qty	Price
1	1	Bakewell Tart	20	0.15
2	1	Danish Pastry	13	0.20
2	4	Danish Pastry	50	0.20
3	1	Apple Pie	45	0.15
3	3	Apple Pie	130	0.15
3	4	Apple Pie	15	0.15
4	2	Butteries	120	0.20
5	3	Steak Pie	30	0.50
6	3	Meringue Pie	20	0.20
7	3	Cherry Pie	15	0.15

c. Derive 2NF

1. Analysis

- The second paradigm requires that the non-primary attributes are fully dependent on the primary key and not partially dependent. In the order table, the total price of an order may depend on both the order number and the quantity and unit price of the items in the order item, there is a partial dependency and further splitting is required.

2. Steps

- Make adjustments to the Orders table to put the calculation of the total order price into the Order Item table.

- Orders table (Orders):

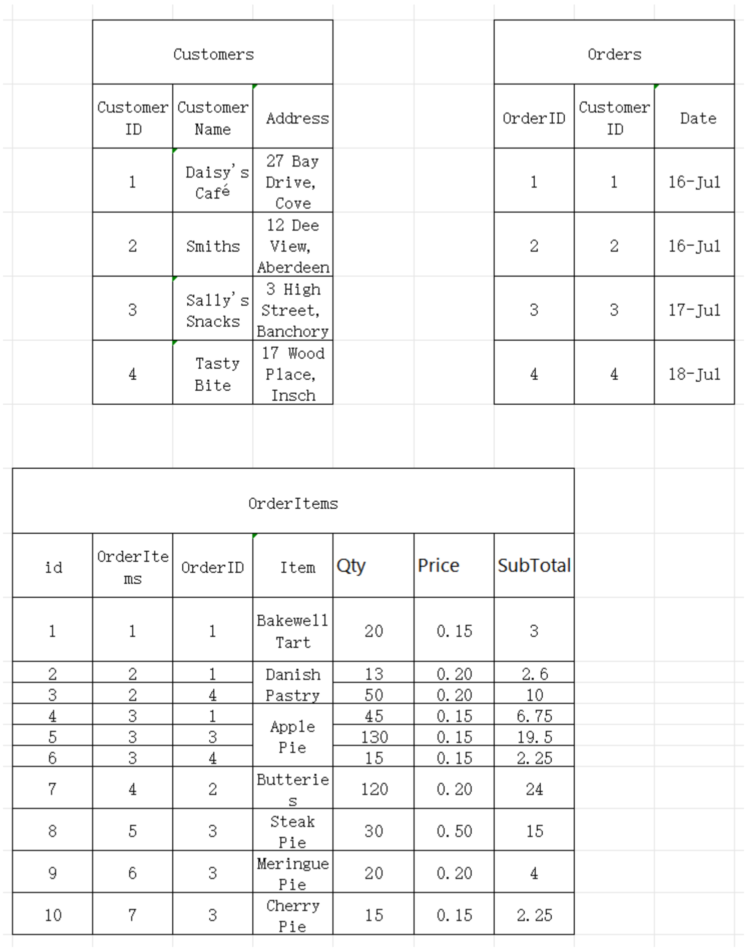
  - Columns: order number (OrderID, can be incremented as the primary key), customer number (CustomerID, as a foreign key associated with the customer table), order date (Date)

- OrderItems table:

  - Primary key: id (can be incremented)

  - foreign key: order number (OrderID)

- Columns: order item number (OrderItemID) - commodity name (Item), quantity (Qty), unit price (Price), the total price of the order item (SubTotal, quantity × unit price)



Customers		
Customer ID	Customer Name	Address
1	Daisy’s Café	27 Bay Drive, Cove
2	Smiths	12 Dee View, Aberdeen
3	Sally’s Snacks	3 High Street, Banchory
4	Tasty Bite	17 Wood Place, Insch

Orders		
OrderID	Customer ID	Date
1	1	16-Jul
2	2	16-Jul
3	3	17-Jul
4	4	18-Jul

OrderItems						
id	OrderItems	OrderID	Item	Qty	Price	SubTotal
1	1	1	Bakewell Tart	20	0.15	3
2	2	1	Danish Pastry	13	0.20	2.6
3	2	4		50	0.20	10
4	3	1	Apple Pie	45	0.15	6.75
5	3	3		130	0.15	19.5
6	3	4		15	0.15	2.25
7	4	2	Butteries	120	0.20	24
8	5	3	Steak Pie	30	0.50	15
9	6	3	Meringue Pie	20	0.20	4
10	7	3	Cherry Pie	15	0.15	2.25

d. Derive 3NF

1. Analysis

The third paradigm requires that non-primary attributes are not passed dependent on the primary key. Observing the adjusted table structure, the third paradigm requirement is now satisfied and there is no passing dependency.

2. Conclusion

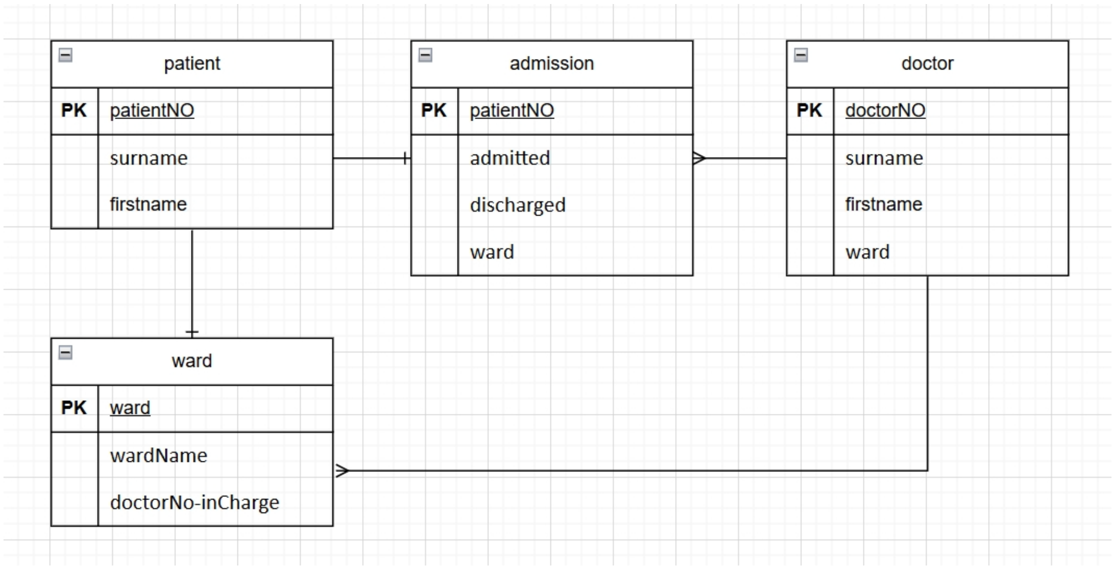
The final three table structures: Customers, Orders and OrderItems meet the third paradigm requirements and can effectively store the bakery's customer information and order data, reducing data redundancy and anomalies.

6.

a. Patient (**patientNo**， **surname**， **firstName**)
Admission (**patientNo**, **admitted**, **discharged**, **ward**)
Doctor (**doctorNo**, **surname**, **firstName**, **ward**)
Ward (**ward**, **wardName**, **doctorNo-inCharge**)

b. a relationship is established between **Patient** and **Doctor** through the **Admission** table, while **Admission** & **Doctor** are directly connected with **Ward** through (the foreign key) ward. **Patient** and **Admission** are directly connected through patientNo.

c.



7.

Table 1: Catering Order

a. Identifying Function Dependencies

1. Order No. determines other attributes

Order Number (Order No.) determines the Customer Number (Account No.), Address (Address), Date (Date), and all order item information in the entire order (including the item name Item, quantity Quantity, and price Price for each order item).

2. Customer number partially determines the address

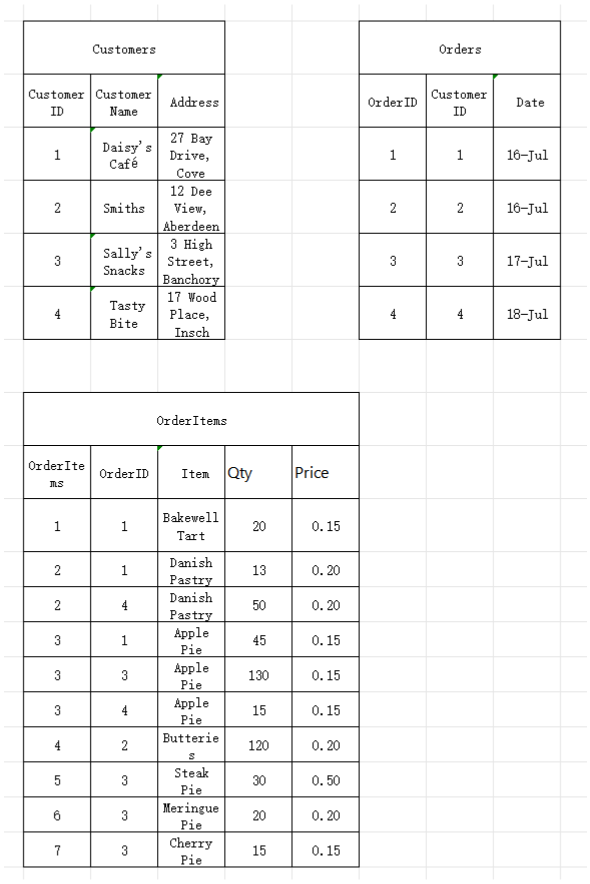
Customer Number (Account No.) and Order Number (Order No.) together to determine the address (Address), because the same customer under different orders may have the same address, but the customer number alone can not fully determine the address, but also need to be combined with the order number (for example, the customer may have more than one shipping address corresponding to different orders).

b. Derivation Paradigms

1. Derivation of 1NF

- ANALYSIS: The original table basically meets the first paradigm requirements, each field is atomic, but there are data redundancy and some function dependency issues.

- Steps
  - Create three tables: Customers, Orders, and OrderItems.
  - Customers table:
    - Primary key: Customer No. (Account No.)
    - Columns: customer name (Customer, can be extracted from the original table), address (Address)
  - Orders table (Orders):
    - Primary key: Order Number (Order No.)
    - Foreign key: Customer No. (Account No.)
    - Columns: order date (Date)
  - OrderItems table (OrderItems):
    - Primary key: OrderItemNo. (can be incremented, such as ItemID)
    - Foreign key: Order No.
    - Columns: commodity name (Item), quantity (Quantity), price (Price)



Customers		
Customer ID	Customer Name	Address
1	Daisy’s Café	27 Bay Drive, Cove
2	Smiths	12 Dee View, Aberdeen
3	Sally’s Snacks	3 High Street, Banchory
4	Tasty Bite	17 Wood Place, Insch

Orders		
OrderID	Customer ID	Date
1	1	16-Jul
2	2	16-Jul
3	3	17-Jul
4	4	18-Jul

OrderItems				
OrderItems	OrderID	Item	Qty	Price
1	1	Bakewell Tart	20	0.15
2	1	Danish Pastry	13	0.20
2	4	Danish Pastry	50	0.20
3	1	Apple Pie	45	0.15
3	3	Apple Pie	130	0.15
3	4	Apple Pie	15	0.15
4	2	Butteries	120	0.20
5	3	Steak Pie	30	0.50
6	3	Meringue Pie	20	0.20
7	3	Cherry Pie	15	0.15

2. Derive 2NF

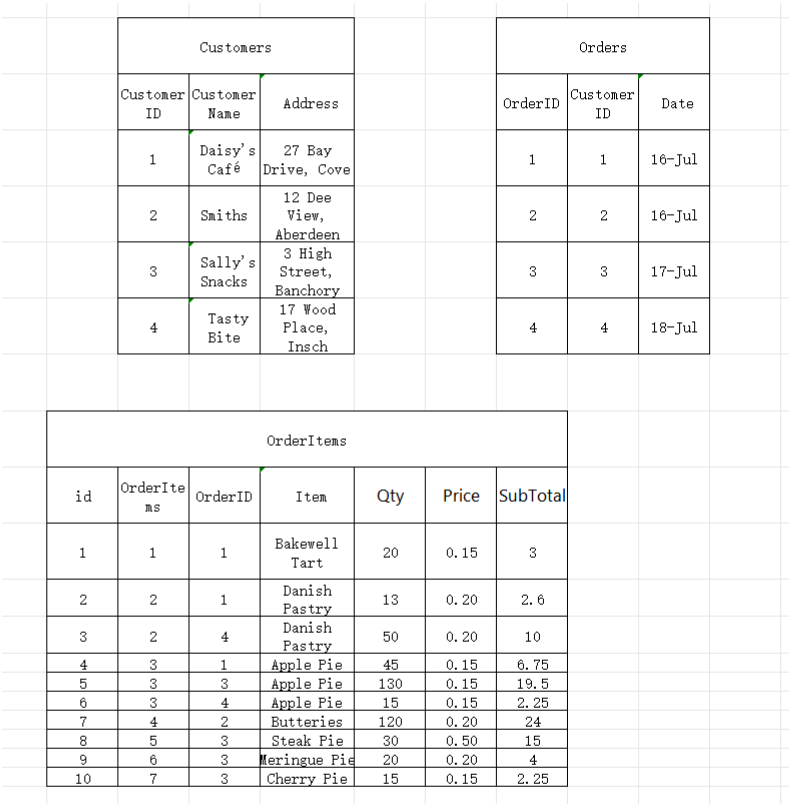
Analysis

- The second paradigm requires that the non-primary attributes are fully dependent on the primary key and not partially dependent. In the order table, the total price of an order may depend on both the order number and the quantity and unit price of the items in the order item, there is

a partial dependency and further splitting is required.
Steps

- Make adjustments to the Orders table to put the calculation of the total order price into the Order Item table.
- Orders table (Orders):
  - Columns: order number (OrderID, can be incremented as the primary key), customer number (CustomerID, as a foreign key associated with the customer table), order date (Date)

- OrderItems table:
  - Primary key: id (can be incremented)
  - foreign key: order number (OrderID)
  - Columns: order item number (OrderItemID) - commodity name (Item), quantity (Qty), unit price (Price), the total price of the order item (SubTotal, quantity × unit price)



Customers		
Customer ID	Customer Name	Address
1	Daisy's Café	27 Bay Drive, Cove
2	Smiths	12 Dee View, Aberdeen
3	Sally's Snacks	3 High Street, Banchory
4	Tasty Bite	17 Wood Place, Insch

Orders		
OrderID	Customer ID	Date
1	1	16-Jul
2	2	16-Jul
3	3	17-Jul
4	4	18-Jul

OrderItems						
id	OrderItems	OrderID	Item	Qty	Price	SubTotal
1	1	1	Bakewell Tart	20	0.15	3
2	2	1	Danish Pastry	13	0.20	2.6
3	2	4	Danish Pastry	50	0.20	10
4	3	1	Apple Pie	45	0.15	6.75
5	3	3	Apple Pie	130	0.15	19.5
6	3	4	Apple Pie	15	0.15	2.25
7	4	2	Butteries	120	0.20	24
8	5	3	Steak Pie	30	0.50	15
9	6	3	Meringue Pie	20	0.20	4
10	7	3	Cherry Pie	15	0.15	2.25

Derivation of 3NF

- Analysis: The three tables adjusted by 2NF do not have pass-through dependencies.
- CONCLUSION: The Customers, Orders and OrderItems tables have met the 3NF requirements.

Table 2: Student records

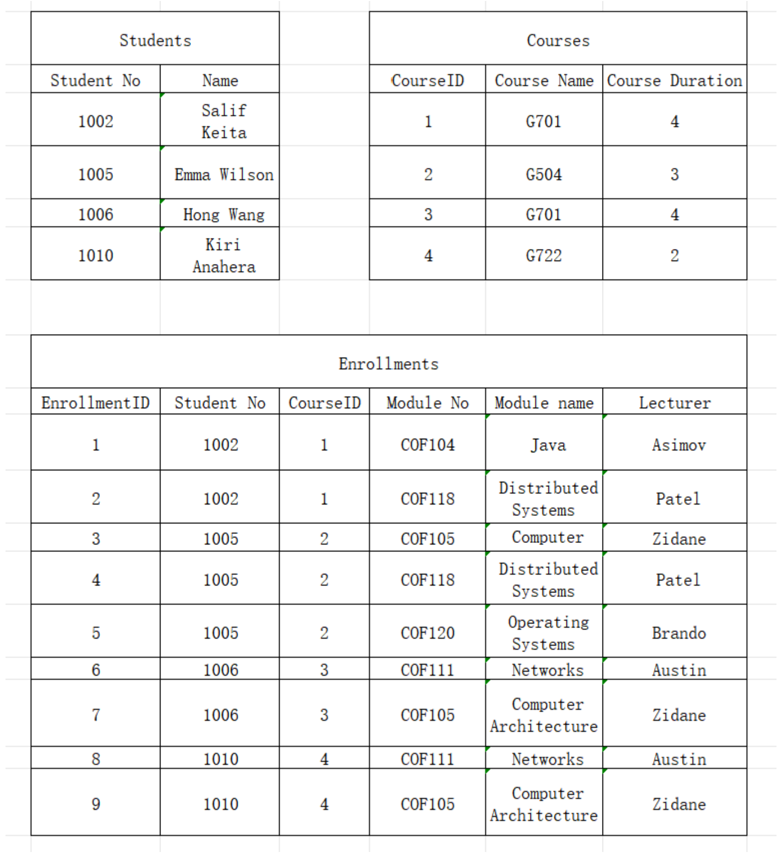
a. Identifying Function Dependencies

1. Student No determines the student's Name, Course, and Course Duration. It can be expressed as: Student No → Name, Course, Course Duration.
2. (Student No, Course) together determine the Module No, Module name and Lecturer. That is, (Student No, Course) → Module No, Module name, Lecturer.

b. Derivation Paradigms

1. Derivation of 1NF

- ANALYSIS: The original table basically satisfies the atomicity requirement, but there are data redundancy and some function dependency problems, which need to be decomposed.
- Steps
    - Create three tables: the Students table (Students), the Courses table (Courses) and the Enrollments table (Enrollments).
    - Student table (Students):
        - Primary key: Student No.
        - Columns: Student Name
    - Courses:
        - Primary key: CourseID
        - Columns: Course Name, Course Duration
    - Enrollments:
        - Primary key: EnrollmentID
        - foreign key: student number (Student No), course number (CourseID)
        - Columns: module number (Module No), module name (Module name), lecturer (Lecturer)



Students

Student No	Name
1002	Salif Keita
1005	Emma Wilson
1006	Hong Wang
1010	Kiri Anahera

Courses

CourseID	Course Name	Course Duration
1	G701	4
2	G504	3
3	G701	4
4	G722	2

Enrollments

EnrollmentID	Student No	CourseID	Module No	Module name	Lecturer
1	1002	1	COF104	Java	Asimov
2	1002	1	COF118	Distributed Systems	Patel
3	1005	2	COF105	Computer	Zidane
4	1005	2	COF118	Distributed Systems	Patel
5	1005	2	COF120	Operating Systems	Brando
6	1006	3	COF111	Networks	Austin
7	1006	3	COF105	Computer Architecture	Zidane
8	1010	4	COF111	Networks	Austin
9	1010	4	COF105	Computer Architecture	Zidane

2. Derivation of 2NF

Analysis: There is a partial dependency issue with the module related attributes in the course selection table, as the module number, module name, and instructor are determined by both the student number and course number, and do not rely entirely on the course selection number as the primary key.

3. Derivation of 3NF

- ANALYSIS: There is no passing dependency in the tables after 2NF adjustment.
- Conclusion: The Student table (Students), Course table (Courses) and Enrollments table (Enrollments) have met the 3NF requirements.

8.

a. Identify function dependencies

1. (Report ID, Reporting Period) Determines Other Attributes

Report ID and Reporting Period together determine Branch Code, Branch Name, Supervisor ID and Supervisor Name. It can be expressed as (Report ID, Reporting Period) → Branch Code, Branch Name, Supervisor ID, Supervisor Name.

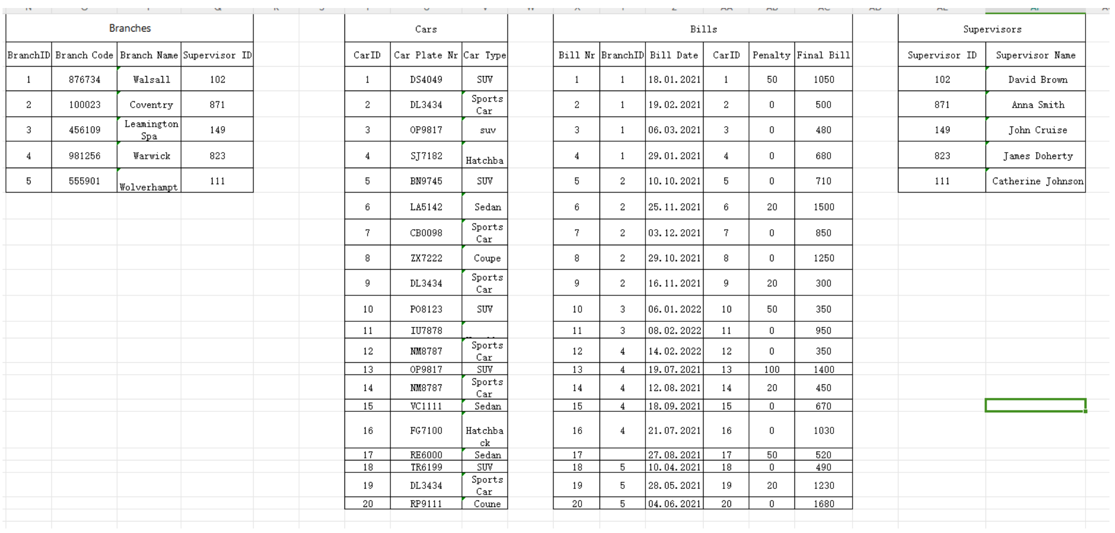
2. (Bill Nr, Bill Date) determines vehicle-related and expense attributes.

Bill Nr and Bill Date together determine Car Plate Nr, Car Type, Penalty and Final Bill. That is, (Bill Nr, Bill Date) → Car Plate Nr, Car Type, Penalty, Final Bill.

b. Derivation Paradigms

1. Derivation of 1NF

- ANALYSIS: The original table suffers from data redundancy and partial function dependency issues and needs to be decomposed to satisfy the first paradigm requirement that each field is atomic and duplicate groups are eliminated.
- Steps
  - Create four tables: Branches, Cars, Bills and Supervisors.
  - Branches table:
    - Primary key: BranchID (can be incremented)
    - Columns: Branch Code, Branch Name, Supervisor ID.
  - Vehicle Table (Cars):
    - Main Key: Vehicle ID (CarID, can be incremented)
    - Columns: Vehicle Plate (Car Plate Nr), Vehicle Type (Car Type)
  - Bills Table (Bills):
    - Primary key: Bill Number (Bill Nr)
    - Foreign key: BranchID.
    - Columns: Bill Date, CarID, Penalty, and Final Bill.
  - Supervisor Table (Supervisors):
    - Primary Key: Supervisor ID
    - Columns: Supervisor Name (Supervisor Name)



Branches

BranchID	Branch Code	Branch Name	Supervisor ID
1	876734	Walsall	102
2	100023	Coventry	871
3	456109	Leamington Spa	149
4	981256	Warwick	823
5	555901	Wolverhampt	111

Cars

CarID	Car Plate Nr	Car Type
1	DS4049	SUV
2	DL3434	Sports Car
3	OP9817	suv
4	SJ7182	Hatchba
5	BN9745	SUV
6	LA5142	Sedan
7	CB0098	Sports Car
8	ZX7222	Coupe
9	DL3434	Sports Car
10	PO8123	SUV
11	IU7878	
12	NM8787	Sports Car
13	OP9817	SUV
14	NM8787	Sports Car
15	VC1111	Sedan
16	FG7100	Hatchback
17	RE6000	Sedan
18	TR6199	SUV
19	DL3434	Sports Car
20	RP9111	Coune

Bills

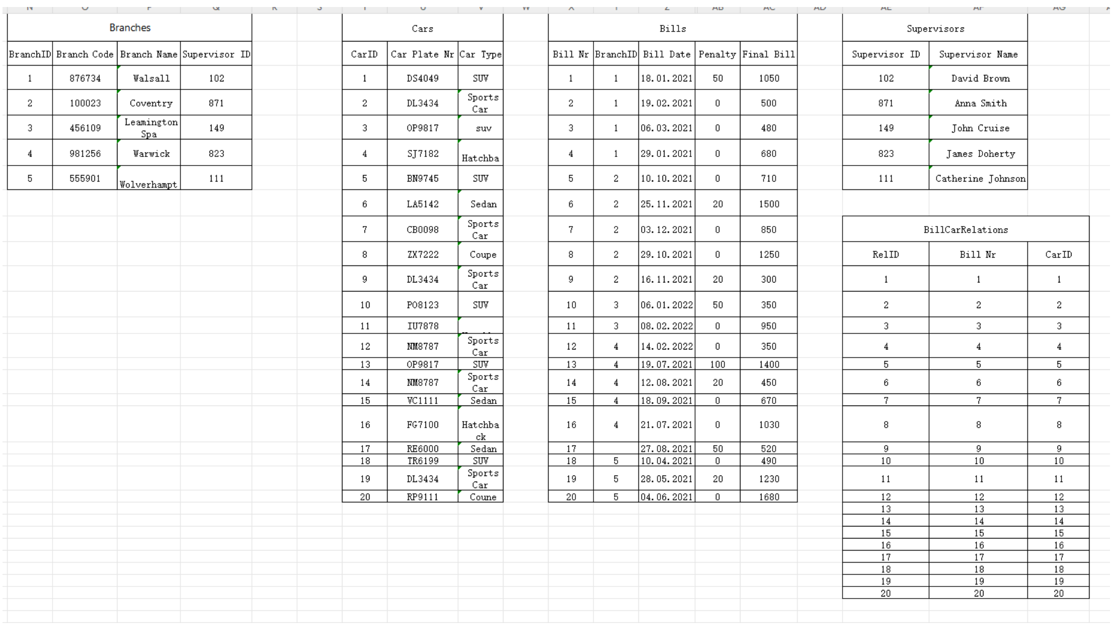
Bill Nr	BranchID	Bill Date	CarID	Penalty	Final Bill
1	1	18.01.2021	1	50	1050
2	1	19.02.2021	2	0	500
3	1	06.03.2021	3	0	480
4	1	29.01.2021	4	0	680
5	2	10.10.2021	5	0	710
6	2	25.11.2021	6	20	1500
7	2	03.12.2021	7	0	850
8	2	29.10.2021	8	0	1250
9	2	16.11.2021	9	20	300
10	3	06.01.2022	10	50	350
11	3	08.02.2022	11	0	950
12	4	14.02.2022	12	0	350
13	4	19.07.2021	13	100	1400
14	4	12.08.2021	14	20	450
15	4	18.09.2021	15	0	670
16	4	21.07.2021	16	0	1030
17		27.08.2021	17	50	520
18	5	10.04.2021	18	0	490
19	5	28.05.2021	19	20	1230
20	5	04.06.2021	20	0	1680

Supervisors

Supervisor ID	Supervisor Name
102	David Brown
871	Anna Smith
149	John Cruise
823	James Doherty
111	Catherine Johnson

2. Derive 2NF

- Analysis: There is a partial dependency problem with the vehicle information in the bill table because the vehicle number is not completely dependent on the primary key of the bill number, but rather determines the vehicle-related information together with the date, which does not meet the second paradigm requirements and needs to be adjusted.
- Steps
  - Adjust the structure of the Bills table.
  - Bills table (Bills):
    - Primary key: Bill Number (Bill Nr)
    - Foreign Key: BranchID
    - Columns: Bill Date (Bill Date), Penalty (Penalty), Final Bill Amount (Final Bill)
  - Create a new association table: BillCarRelations
    - Primary key: Relation ID (RelID, can be incremented)
    - Foreign key: Bill Number (Bill Nr), Vehicle Number (CarID)



Branches

BranchID	Branch Code	Branch Name	Supervisor ID
1	876734	Walsall	102
2	100023	Coventry	871
3	456109	Leamington Spa	149
4	981256	Warwick	823
5	555901	Wolverhampt	111

Cars

CarID	Car Plate Nr	Car Type
1	DS4049	SUV
2	DL3434	Sports Car
3	OP9817	suv
4	SJ7182	Hatchba
5	BN9745	SUV
6	LA5142	Sedan
7	CB0098	Sports Car
8	ZX7222	Coupe
9	DL3434	Sports Car
10	PO8123	SUV
11	IU7878	
12	NM8787	Sports Car
13	OP9817	SUV
14	NM8787	Sports Car
15	VC1111	Sedan
16	FG7100	Hatchback
17	RE6000	Sedan
18	TR6199	SUV
19	DL3434	Sports Car
20	RP9111	Coune

Bills

Bill Nr	BranchID	Bill Date	Penalty	Final Bill
1	1	18.01.2021	50	1050
2	1	19.02.2021	0	500
3	1	06.03.2021	0	480
4	1	29.01.2021	0	680
5	2	10.10.2021	0	710
6	2	25.11.2021	20	1500
7	2	03.12.2021	0	850
8	2	29.10.2021	0	1250
9	2	16.11.2021	20	300
10	3	06.01.2022	50	350
11	3	08.02.2022	0	950
12	4	14.02.2022	0	350
13	4	19.07.2021	100	1400
14	4	12.08.2021	20	450
15	4	18.09.2021	0	670
16	4	21.07.2021	0	1030
17		27.08.2021	50	520
18	5	10.04.2021	0	490
19	5	28.05.2021	20	1230
20	5	04.06.2021	0	1680

Supervisors

Supervisor ID	Supervisor Name
102	David Brown
871	Anna Smith
149	John Cruise
823	James Doherty
111	Catherine Johnson

BillCarRelations

RelID	Bill Nr	CarID
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10
11	11	11
12	12	12
13	13	13
14	14	14
15	15	15
16	16	16
17	17	17
18	18	18
19	19	19
20	20	20

3. Derivation of 3NF

- ANALYSIS: The tables adjusted for 2NF do not have pass-through dependencies.
- Conclusion: The Branches table (Branches), the Vehicles table (Cars), the Bills table (Bills), the Supervisors table (Supervisors), and the BillCarRelations table (BillCarRelations) have met the 3NF requirements.

9.(1)

Followings are tables for **maintenance**, **engineer**, **boat**, **contractor**, **task,** for the M70 scenario.

maintenanceID	boatID	engineerID	marine equipment	taskID	time & date	servicing time
M001	B001	E02	scrubbers for inert-gas generator	T03	20 10 2006	43 23
M002	B002	E01	submerged equipment	T03	16 06 2013	16 30
M003	B003	E05	multimedia	T02	05 08 2012	08 26

engineerID	engineerName	expertise	man-hours
E01	gears	machinery	365 43
E02	clef	material	289 31
E03	kondraki	optics	230 21
E04	bright	marine ecology	1854 53
E05	kain	machinery & marine ecology	210 46
E06	king	mathematics	223 45
E07	iceberg	statistic	276 22

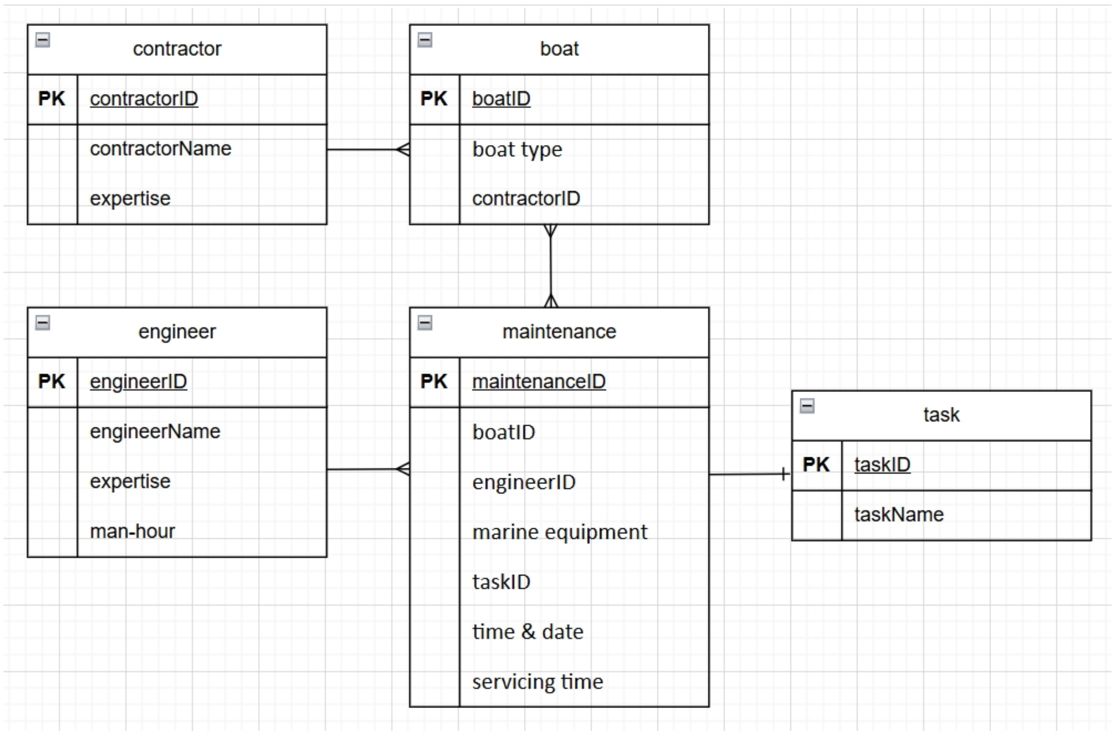
boatID	boat type	contractorID
B001	freighter	C001
B002	freighter	C002
B003	yacht	C003

contractorID	contractorName	expertise
C001	goc	marine
C002	urban	machine
C003	fufu	astronomy

taskID	taskName
T01	Service
T02	Software upgrade
T03	Repair
T04	Safety inspection
T05	Other (customer specific requests)

(2)

Below is the corresponding **ER diagram** for the M70 scenario.


contractor
PK contractorID
contractorName
expertise
boat
PK boatID
boat type
contractorID
engineer
PK engineerID
engineerName
expertise
man-hour
maintenance
PK maintenanceID
boatID
engineerID
marine equipment
taskID
time & date
servicing time
task
PK taskID
taskName