KONBININFO

A Convenience Store Inventory Database System

A Project Presentation

by RainbowDB

Cogaed · **Del Castillo** · **Kerchaten** · **Vanzuela** ⁂ CC5·2A ⁂ 3 December 2025

JDRC

Project Overview

- **Domain:** Convenience Store Inventory System
- **Project Objectives:**
  - To create an **organised** , **normalised** , **easily usable** , and **easily maintainable** inventory database system for convenience stores
  - To **digitalise the cataloguing** of products in stock in order to be able to accurately track stock levels

Phase 1

Domain Analysis & Data Structuring

Summary of Progress

WHAT WE DID:

- Identified the **entities** and their **attributes**
- Identified the proper **data types** for their attributes, to ensure clarity, consistency, and scalability
- Laid the foundations for ERD modelling and future SQL implementation

Entities 1/2

# = Primary Key

## Product

Attribute	Data Type
# Product ID	integer
Product Supplier	integer
Product Category	integer
Product Serial Number	varchar(64)
Product Name	varchar(64)
Product Description	text
Product Arrival Date	datetime
Product Expiry Date	datetime
Product Price per Unit	decimal
Product Stock Quantity	integer

## Product Category

Attribute	Data Type
# Product Category ID	integer
Product Category Name	varchar(64)
Product Category Description	text

## Shelf

Attribute	Data Type
# Shelf ID	integer
Shelf Name	varchar(64)
Shelf Type	varchar(64)
Shelf Capacity	integer

Entities 2/2

# = Primary Key

Supplier

Attribute	Data
Supplier ID	integer
Supplier Name	varchar(64)
Supplier Address	varchar(255)
Supplier Schedule	text

Manufacturer

Attribute	Data
Manufacturer ID	integer
Manufacturer Name	varchar(64)
Manufacturer Address	varchar(255)
Manufacturer Description	text

Equipment

Attribute	Data Type
Equipment ID	integer
Equipment Compatible Product Categories *(will be replaced by join table)*	integer
Equipment Manufacturer	integer
Equipment Name	varchar(64)
Equipment Description	text
Equipment Maintenance	text
Equipment Serial Number	varchar(64)
Equipment Arrival Date	datetime
Equipment Energy Rating	decimal

Phase 2

Entity Modeling & Schema Planning

Summary of Progress

This part focused on entity modeling and planning.
It included:

- Assigning **data types**
- Defining entity **relationships** (in the ERD)
- Designing the database **structure**
- Defining **Business Rules**
- Deciding which diagramming and collaboration tools to use

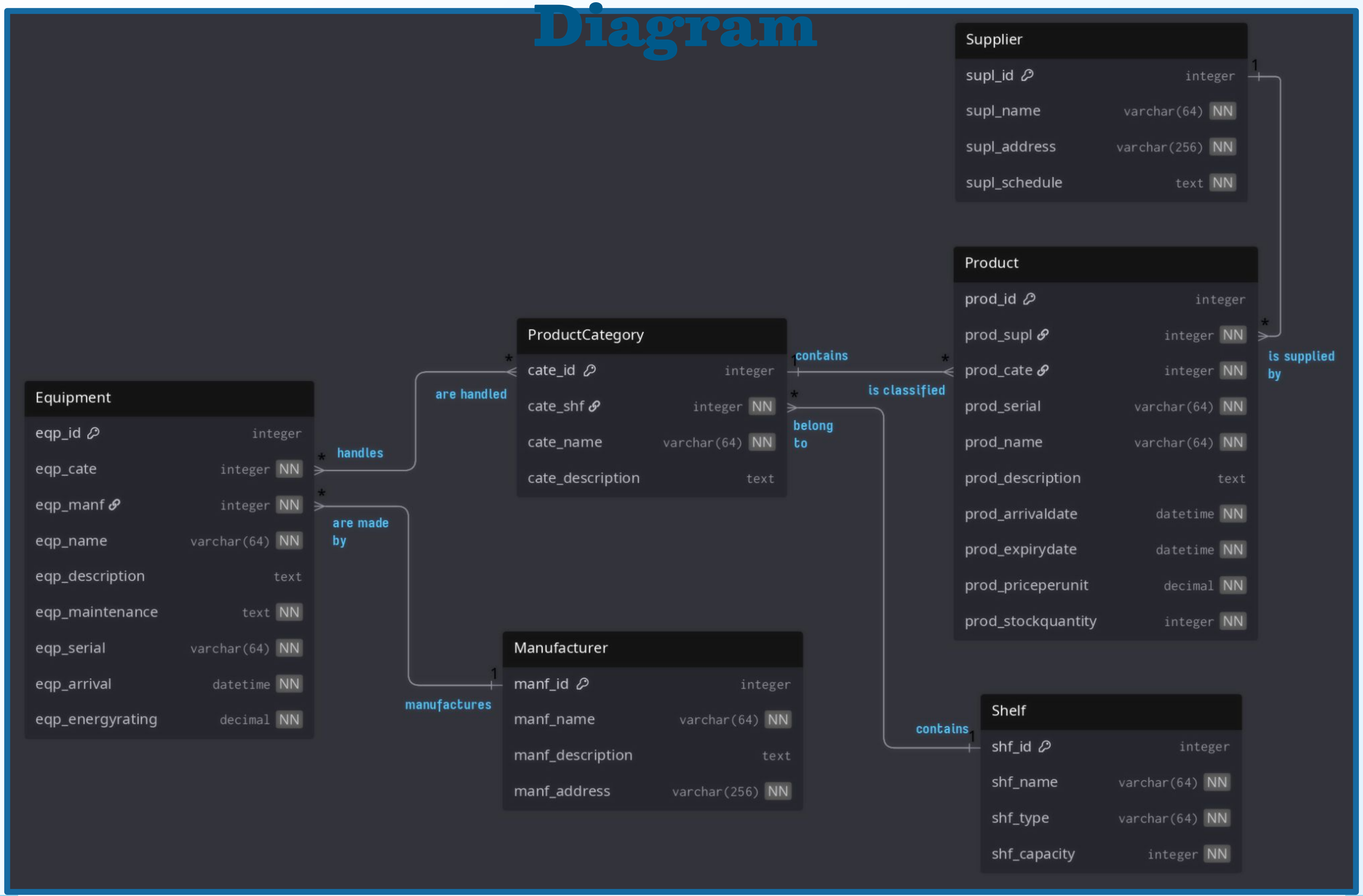
Business Rules 1/2

- A **Product** needs to be assigned to a **Supplier** .
- A **Supplier** can be assigned to one or more **Products** .
- A **Product** needs to be assigned to a **Product Category** .
- A **Product Category** can hold one or more **Products** .
- A **Product Category** needs to be assigned to a **Shelf** .

Business Rules 2/2

- A **Shelf** can hold one or more **Product Categories** .
- An **Equipment** can be compatible with one or more **Product Categories** .
- An **Equipment** must be assigned to one **Manufacturer** .
- A **Manufacturer** can make one or more **Equipments** .

Good Enough Entity Relational Diagram



Supplier	
supl_id 🔑	integer
supl_name	varchar(64) NN
supl_address	varchar(256) NN
supl_schedule	text NN

Product	
prod_id 🔑	integer
prod_supl 🔗	integer NN
prod_cate 🔗	integer NN
prod_serial	varchar(64) NN
prod_name	varchar(64) NN
prod_description	text
prod_arrivaldate	datetime NN
prod_expirydate	datetime NN
prod_priceperunit	decimal NN
prod_stockquantity	integer NN

ProductCategory	
cate_id 🔑	integer
cate_shf 🔗	integer NN
cate_name	varchar(64) NN
cate_description	text

Equipment	
eqp_id 🔑	integer
eqp_cate	integer NN
eqp_manf 🔗	integer NN
eqp_name	varchar(64) NN
eqp_description	text
eqp_maintenance	text NN
eqp_serial	varchar(64) NN
eqp_arrival	datetime NN
eqp_energyrating	decimal NN

Manufacturer	
manf_id 🔑	integer
manf_name	varchar(64) NN
manf_description	text
manf_address	varchar(256) NN

Shelf	
shf_id 🔑	integer
shf_name	varchar(64) NN
shf_type	varchar(64) NN
shf_capacity	integer NN

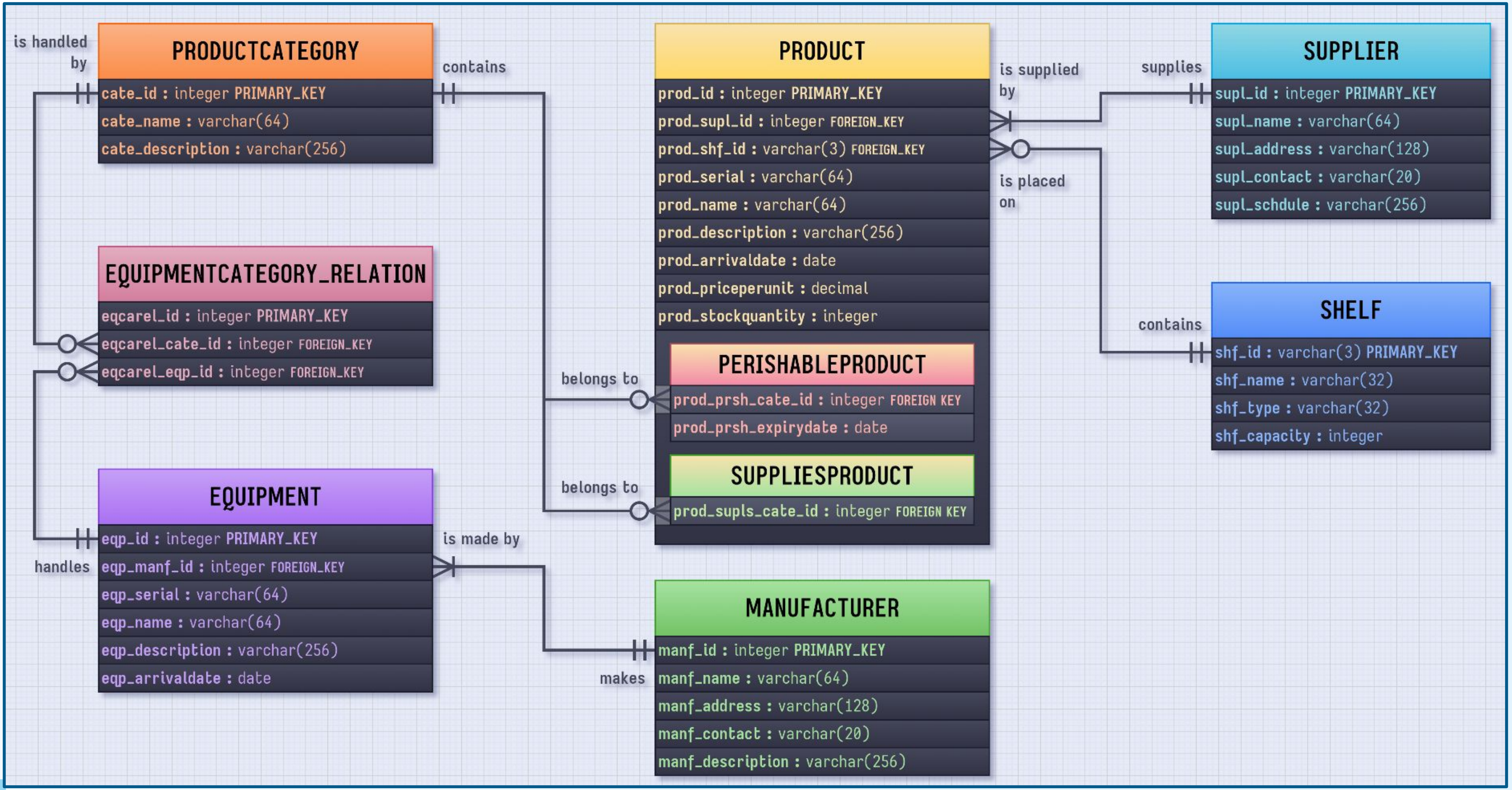
Phase 3

Advanced ERD & Relationships with SQL

Summary of Progress

- **Refined** the ERD; implemented intersection entity and entity subtype
- **Conversion** of business rules to ERD relationships
- Implemented **SQL schema**
- Inserted **sample data** into each table
- **Tested several SQL queries** to demonstrate the functionality of the database

Better Entity Relational Diagram



PRODUCTCATEGORY
cate_id : integer PRIMARY_KEY
cate_name : varchar(64)
cate_description : varchar(256)

EQUIPMENTCATEGORY_RELATION
eqcarel_id : integer PRIMARY_KEY
eqcarel_cate_id : integer FOREIGN_KEY
eqcarel_eqp_id : integer FOREIGN_KEY

EQUIPMENT
eqp_id : integer PRIMARY_KEY
eqp_manf_id : integer FOREIGN_KEY
eqp_serial : varchar(64)
eqp_name : varchar(64)
eqp_description : varchar(256)
eqp_arrivaldate : date

PRODUCT
prod_id : integer PRIMARY_KEY
prod_supl_id : integer FOREIGN_KEY
prod_shf_id : varchar(3) FOREIGN_KEY
prod_serial : varchar(64)
prod_name : varchar(64)
prod_description : varchar(256)
prod_arrivaldate : date
prod_priceperunit : decimal
prod_stockquantity : integer

PERISHABLEPRODUCT
prod_prsh_cate_id : integer FOREIGN KEY
prod_prsh_expirydate : date

SUPPLIESPRODUCT
prod_supls_cate_id : integer FOREIGN KEY

MANUFACTURER
manf_id : integer PRIMARY_KEY
manf_name : varchar(64)
manf_address : varchar(128)
manf_contact : varchar(20)
manf_description : varchar(256)

SUPPLIER
supl_id : integer PRIMARY_KEY
supl_name : varchar(64)
supl_address : varchar(128)
supl_contact : varchar(20)
supl_schdule : varchar(256)

SHELF
shf_id : varchar(3) PRIMARY_KEY
shf_name : varchar(32)
shf_type : varchar(32)
shf_capacity : integer

SQL Scripts 1/3

Product Table

```sql
CREATE TABLE `Product` (
    `prod_id` int PRIMARY KEY,
    `prod_supl_id` int NOT NULL,
    `prod_shf_id` varchar(3),
    `prod_serial` varchar(64),
    `prod_name` varchar(64),
    `prod_description` varchar(256),
    `prod_arrivaldate` date,
    `prod_priceperunit` decimal,
    `prod_stockqty` int,
    `prod_prsh_cate_id` int,
    `prod_prsh_expirydate` date,
    `prod_supls_cate_id` int
);
```

Supplier Table

```sql
CREATE TABLE `Supplier` (
    `supl_id` int PRIMARY KEY,
    `supl_name` varchar(64) NOT NULL,
    `supl_contact` varchar(20) NOT NULL,
    `supl_address` varchar(128) NOT NULL,
    `supl_schedule` varchar(128) NOT NULL,
);
```

Shelf Table

```sql
CREATE TABLE `Shelf` (
    `shf_id` int PRIMARY KEY,
    `shf_name` varchar(32),
    `shf_type` varchar(32),
    `shf_capacity` int
);
```

SQL Scripts 2/3

Product Category Table

```
CREATE TABLE `ProductCategory` (
    `cate_id` int PRIMARY KEY,
    `cate_name` varchar(64),
    `cate_description` varchar(256),
 );
```

Manufacturer Table

```
CREATE TABLE `Manufacturer` (
    `manf_id` int PRIMARY KEY,
    `manf_name` varchar(64),
    `manf_contact` varchar(20) NOT NULL,
    `manf_address` varchar(128),
    `manf_description` varchar(128)
);
```

Equipment Table

```
CREATE TABLE `Equipment` (
    `eqp_id` int PRIMARY KEY,
    `eqp__manf_id` int NOT NULL,
    `eqp_serial` varchar(64),
    `eqp_name` varchar(64),
    `eqp_description` varchar(256),
    `eqp_arrivaldate` date
);
```

Equipment-Category Relation Table

```
CREATE TABLE
     `EquipmentCategoryRelation` (
    `eqcarel_id` int PRIMARY KEY,
    `eqcarel_cate_id` int,
    `eqcarel_eqp_id` int
 );
```

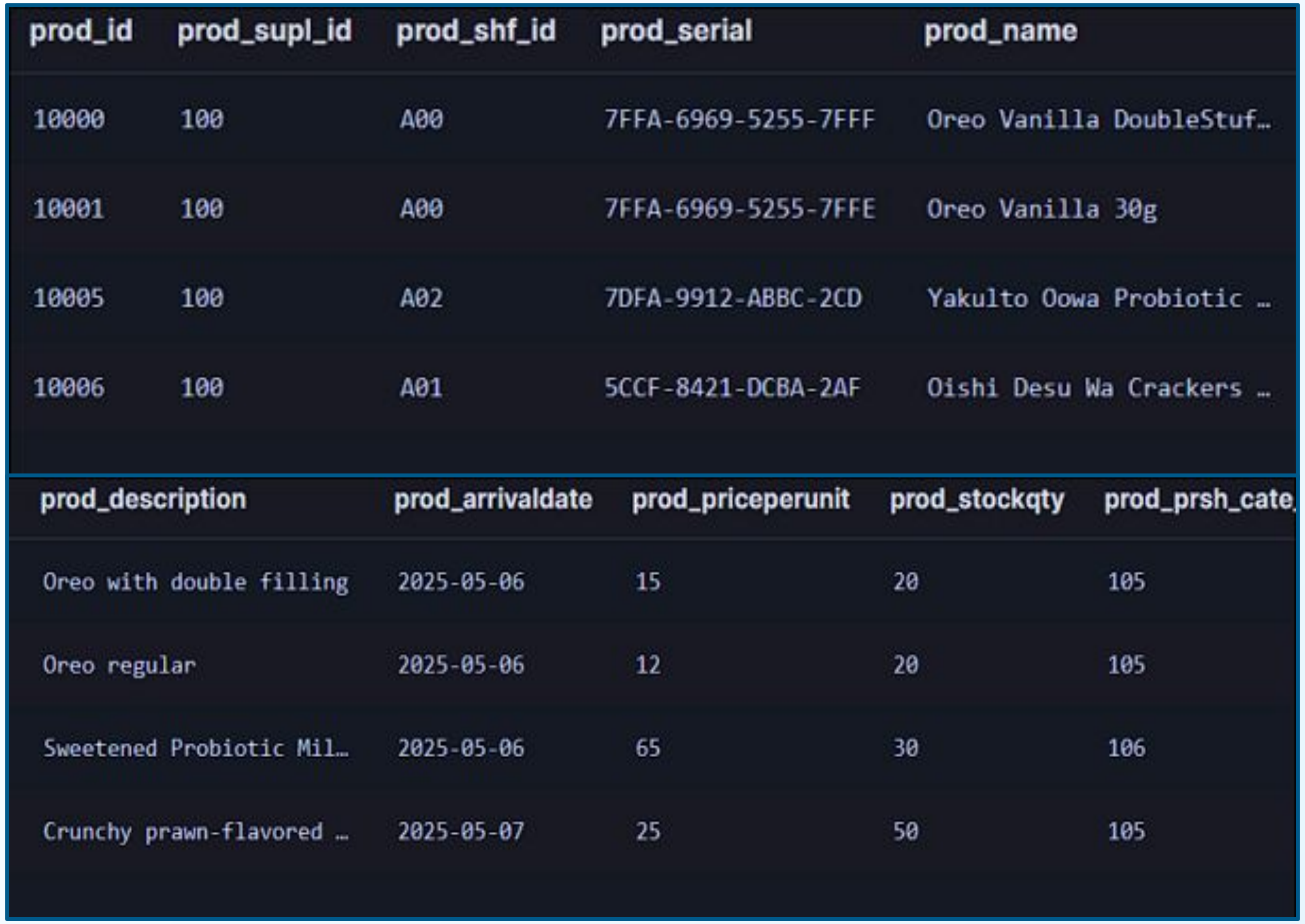
SQL Scripts 3/3

Foreign Keys

```
ALTER TABLE `Product` ADD FOREIGN KEY (`prod_supl_id`) REFERENCES `Supplier`
(`supl_id`);

ALTER TABLE `Product` ADD FOREIGN KEY (`prod_shf_id`) REFERENCES `Shelf`
(`shf_id`);

ALTER TABLE `Product` ADD FOREIGN KEY (`prod_prsh_cate_id`) REFERENCES
`ProductCategory` (`cate_id`);

ALTER TABLE `Product` ADD FOREIGN KEY (`prod_supls_cate_id`) REFERENCES
`ProductCategory` (`cate_id`);

ALTER TABLE `EquipmentCategoryRelation` ADD FOREIGN KEY (`eqcarel_cate_id`)
REFERENCES `ProductCategory` (`cate_id`);

ALTER TABLE `EquipmentCategoryRelation` ADD FOREIGN KEY (`eqcarel_eqp_id`)
REFERENCES `Equipment` (`eqp_id`);

ALTER TABLE `Equipment` ADD FOREIGN KEY (`eqp_manf_id`) REFERENCES `Manufacturer`
(`manf_id`);
```

SQL Query #1

- SELECT * FROM Product WHERE prod_shf_id LIKE 'A%’



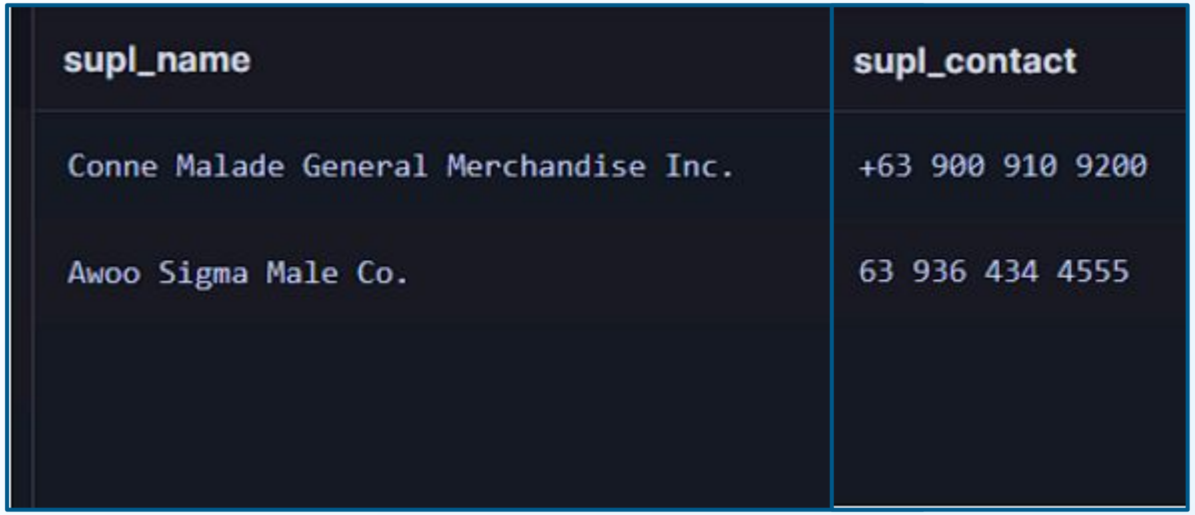
prod_id	prod_supl_id	prod_shf_id	prod_serial	prod_name	prod_description	prod_arrivaldate	prod_priceperunit	prod_stockqty	prod_prsh_cate
10000	100	A00	7FFA-6969-5255-7FFF	Oreo Vanilla DoubleStuf…	Oreo with double filling	2025-05-06	15	20	105
10001	100	A00	7FFA-6969-5255-7FFE	Oreo Vanilla 30g	Oreo regular	2025-05-06	12	20	105
10005	100	A02	7DFA-9912-ABBC-2CD	Yakulto Oowa Probiotic …	Sweetened Probiotic Mil…	2025-05-06	65	30	106
10006	100	A01	5CCF-8421-DCBA-2AF	Oishi Desu Wa Crackers …	Crunchy prawn-flavored …	2025-05-07	25	50	105

List of all Products on Shelf group A.

- Fetches the lists of all products within the same shelf group: A.
- The **LIKE** keyword is used to filter records based on a specified pattern in a column.

SQL Query #2

- SELECT supl_name, supl_contact FROM Supplier WHERE schedule LIKE '%Monday%' OR supl_schedule LIKE '%Wednesday%'

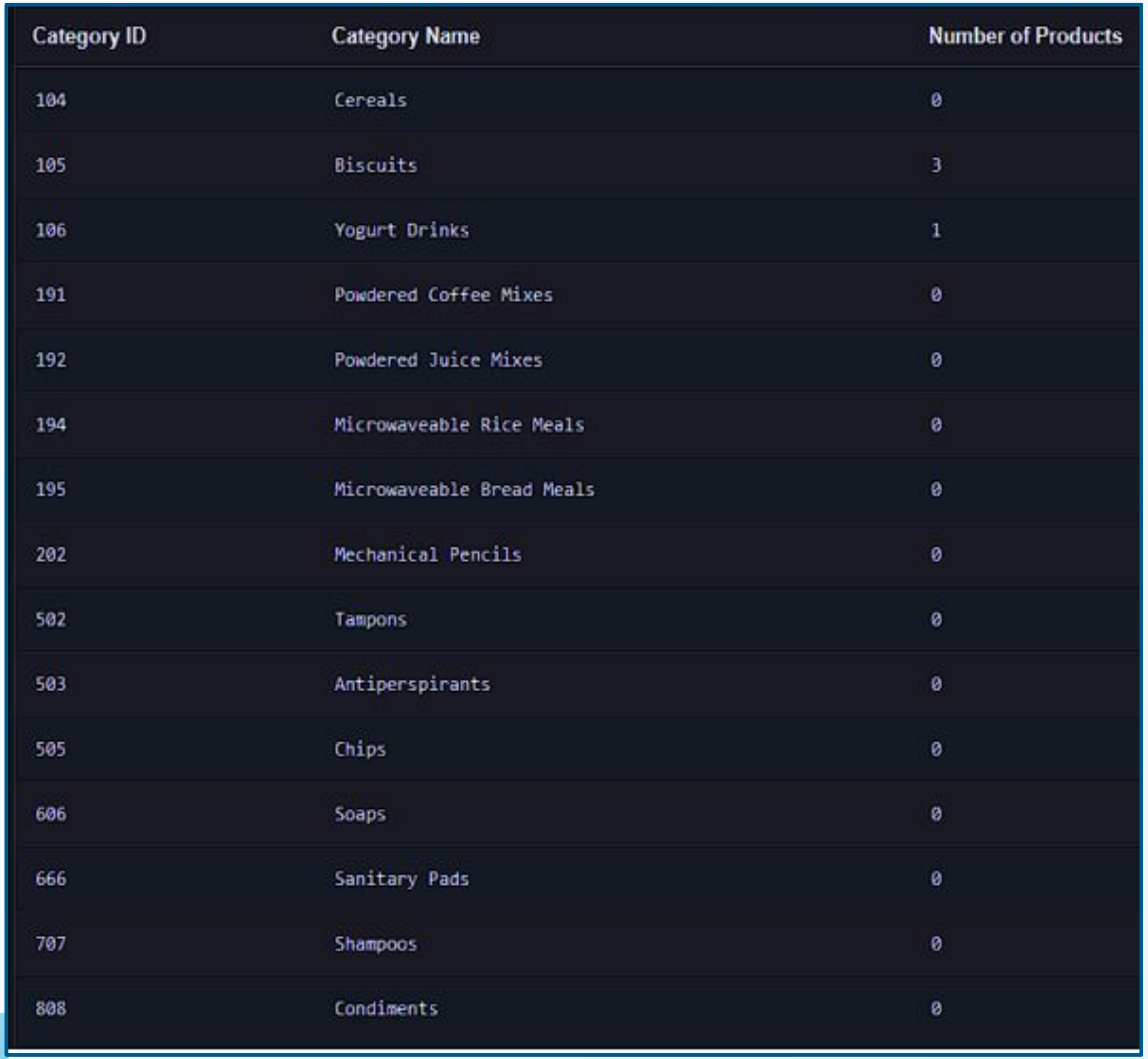


supl_name	supl_contact
Conne Malade General Merchandise Inc.	+63 900 910 9200
Awoo Sigma Male Co.	63 936 434 4555

List all Suppliers that deliver on Mondays or Wednesdays — but *only the name and the contact information)* .

- The **OR** keyword is used to return records based on more than one condition.

SQL Query #3



Category ID	Category Name	Number of Products
104	Cereals	0
105	Biscuits	3
106	Yogurt Drinks	1
191	Powdered Coffee Mixes	0
192	Powdered Juice Mixes	0
194	Microwaveable Rice Meals	0
195	Microwaveable Bread Meals	0
202	Mechanical Pencils	0
502	Tampons	0
503	Antiperspirants	0
505	Chips	0
606	Soaps	0
666	Sanitary Pads	0
707	Shampoos	0
808	Condiments	0

```
SELECT cate_id AS "Category ID", cate_name AS "Category Name",  COUNT(DISTINCT prod_id AS "Number of Products")

FROM ProductCategory

LEFT JOIN Product ON cate_id = prod_prsh_cate_id 5

GROUP BY cate_id, cate_name 6

ORDER by cate_id;
```

Displays all product categories with their IDs and the number of products in each.

- It uses a **LEFT JOIN** to include categories with no perishable products and groups the results by category ID and name.

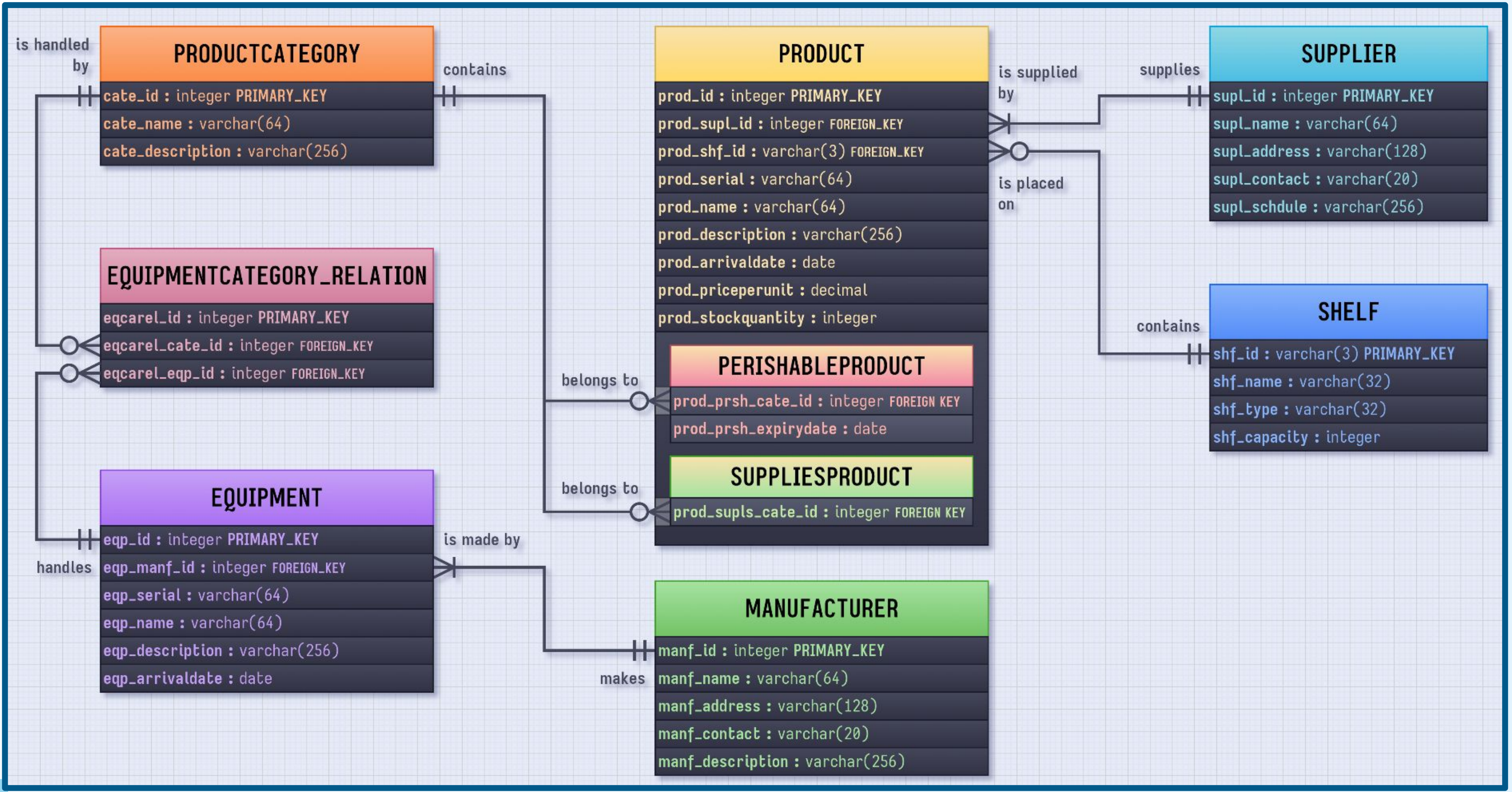
Phase 4

Normalisation & Data Integrity

Summary of Progress

- Added `ShelfType` table to bring schema to 3NF
- Added `InventoryMovement` entity to help track changes in inventory levels
    - Added `InventoryMovementType` entity to categorise those changes and to ensure 3NF compliance
- Checked for 4NF and 5NF violations (none)

Better Entity Relational Diagram



PRODUCTCATEGORY

cate_id : integer PRIMARY_KEY
cate_name : varchar(64)
cate_description : varchar(256)

EQUIPMENTCATEGORY_RELATION

eqcarel_id : integer PRIMARY_KEY
eqcarel_cate_id : integer FOREIGN_KEY
eqcarel_eqp_id : integer FOREIGN_KEY

EQUIPMENT

eqp_id : integer PRIMARY_KEY
eqp_manf_id : integer FOREIGN_KEY
eqp_serial : varchar(64)
eqp_name : varchar(64)
eqp_description : varchar(256)
eqp_arrivaldate : date

PRODUCT

prod_id : integer PRIMARY_KEY
prod_supl_id : integer FOREIGN_KEY
prod_shf_id : varchar(3) FOREIGN_KEY
prod_serial : varchar(64)
prod_name : varchar(64)
prod_description : varchar(256)
prod_arrivaldate : date
prod_priceperunit : decimal
prod_stockquantity : integer

PERISHABLEPRODUCT

prod_prsh_cate_id : integer FOREIGN KEY
prod_prsh_expirydate : date

SUPPLIESPRODUCT

prod_supls_cate_id : integer FOREIGN KEY

MANUFACTURER

manf_id : integer PRIMARY_KEY
manf_name : varchar(64)
manf_address : varchar(128)
manf_contact : varchar(20)
manf_description : varchar(256)

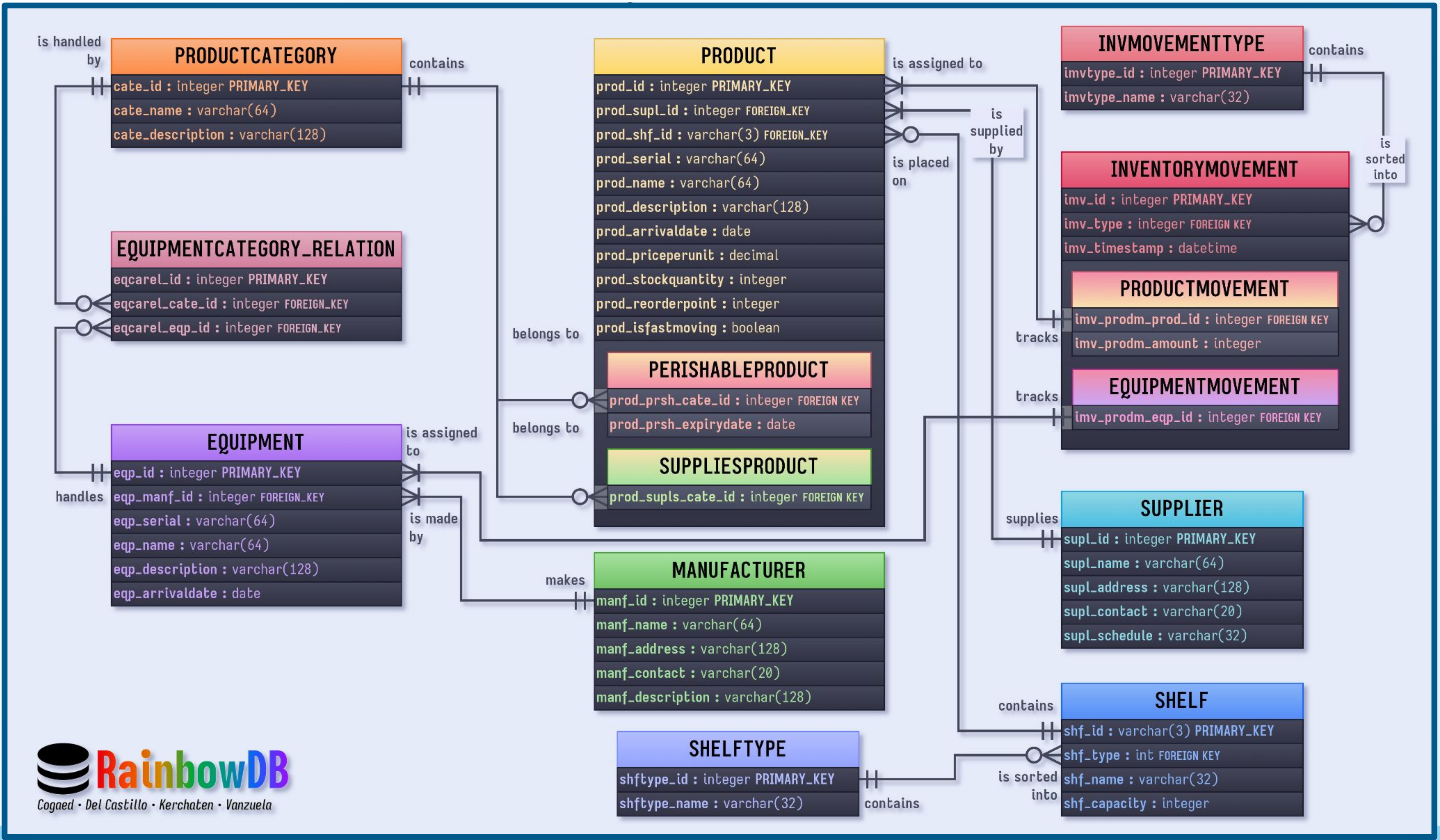
SUPPLIER

supl_id : integer PRIMARY_KEY
supl_name : varchar(64)
supl_address : varchar(128)
supl_contact : varchar(20)
supl_schdule : varchar(256)

SHELF

shf_id : varchar(3) PRIMARY_KEY
shf_name : varchar(32)
shf_type : varchar(32)
shf_capacity : integer

Even Better Entity Relational



PRODUCTCATEGORY
cate_id : integer PRIMARY_KEY
cate_name : varchar(64)
cate_description : varchar(128)

EQUIPMENTCATEGORY_RELATION
eqcarel_id : integer PRIMARY_KEY
eqcarel_cate_id : integer FOREIGN_KEY
eqcarel_eqp_id : integer FOREIGN_KEY

EQUIPMENT
eqp_id : integer PRIMARY_KEY
eqp_manf_id : integer FOREIGN_KEY
eqp_serial : varchar(64)
eqp_name : varchar(64)
eqp_description : varchar(128)
eqp_arrivaldate : date

PRODUCT
prod_id : integer PRIMARY_KEY
prod_supl_id : integer FOREIGN_KEY
prod_shf_id : varchar(3) FOREIGN_KEY
prod_serial : varchar(64)
prod_name : varchar(64)
prod_description : varchar(128)
prod_arrivaldate : date
prod_priceperunit : decimal
prod_stockquantity : integer
prod_reorderpoint : integer
prod_isfastmoving : boolean

PERISHABLEPRODUCT
prod_prsh_cate_id : integer FOREIGN KEY
prod_prsh_expirydate : date

SUPPLIESPRODUCT
prod_supls_cate_id : integer FOREIGN KEY

MANUFACTURER
manf_id : integer PRIMARY_KEY
manf_name : varchar(64)
manf_address : varchar(128)
manf_contact : varchar(20)
manf_description : varchar(128)

SHELFTYPE
shftype_id : integer PRIMARY_KEY
shftype_name : varchar(32)

INVMOVEMENTTYPE
imvtype_id : integer PRIMARY_KEY
imvtype_name : varchar(32)

INVENTORYMOVEMENT
imv_id : integer PRIMARY_KEY
imv_type : integer FOREIGN KEY
imv_timestamp : datetime

PRODUCTMOVEMENT
imv_prodm_prod_id : integer FOREIGN KEY
imv_prodm_amount : integer

EQUIPMENTMOVEMENT
imv_prodm_eqp_id : integer FOREIGN KEY

SUPPLIER
supl_id : integer PRIMARY_KEY
supl_name : varchar(64)
supl_address : varchar(128)
supl_contact : varchar(20)
supl_schedule : varchar(32)

SHELF
shf_id : varchar(3) PRIMARY_KEY
shf_type : int FOREIGN KEY
shf_name : varchar(32)
shf_capacity : integer

Cogaed · Del Castillo · Kerchaten · Vanzuela

Phase 5

Relational Database Design & Query Optimisation

Summary of Progress

- **Added three indexes** for the Product, Supplier, and InventoryMovement tables to optimize lookup, filtering, and grouping operations.
- **Successfully optimised** three SELECT queries, each aligned to use its corresponding index.
- Achieved noticeable **query runtime improvements** .
- **Confirmed that indexes helped** speed up filtering, sorting, and grouping, especially for large datasets.
- **Finalized the optimisation strategy** focusing on creating indexes for frequently searched columns and ensuring SELECT queries effectively use those indexes.

Index Creation

1. **Index for PRODUCT:**

   -
     ```
     CREATE INDEX idx_product_fastmoving_stock
     ON Product(prod_isfastmoving, prod_stockqty);
     ```

2. **Index for SUPPLIER:**

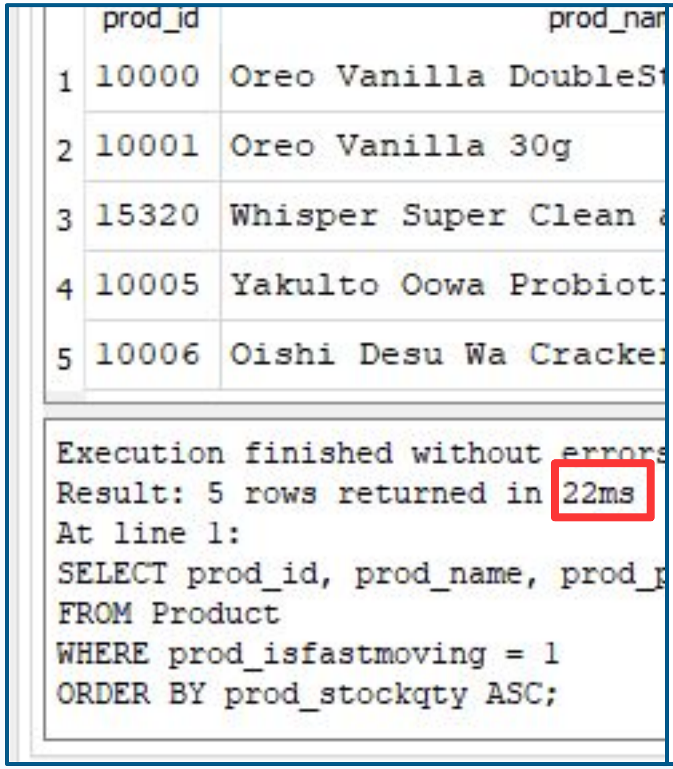
   -
     ```
     CREATE INDEX idx_supl_supl_id
     ON Supplier (supl_id);
     ```

3. **Index for INVENTORYMOVEMENT:**

   -
     ```
     CREATE INDEX idx_imv_type
     ON InventoryMovement(imv_prodm_prod_id) ASC;
     ```

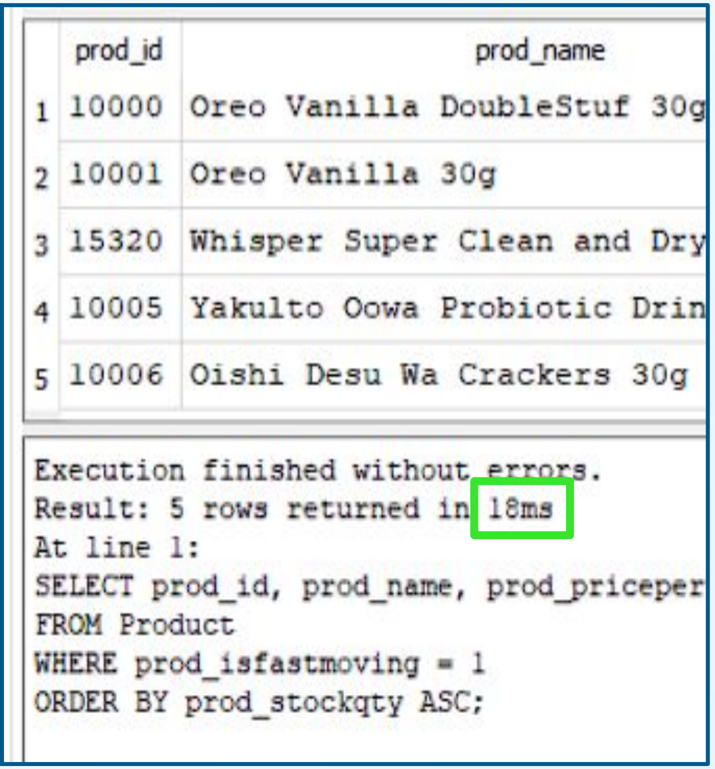
Optimised Queries & Performance Results

1.
```
SELECT prod_id, prod_name, prod_pricceperunit,
prod_stockqty FROM Product WHERE prod_isfastmoving = 1
ORDER BY prod_stockqty ASC;
```



	prod_id	prod_nar
1	10000	Oreo Vanilla DoubleSt
2	10001	Oreo Vanilla 30g
3	15320	Whisper Super Clean a
4	10005	Yakulto Oowa Probiot
5	10006	Oishi Desu Wa Cracke

```
Execution finished without errors
Result: 5 rows returned in 22ms
At line 1:
SELECT prod_id, prod_name, prod_p
FROM Product
WHERE prod_isfastmoving = 1
ORDER BY prod_stockqty ASC;
```



	prod_id	prod_name
1	10000	Oreo Vanilla DoubleStuf 30g
2	10001	Oreo Vanilla 30g
3	15320	Whisper Super Clean and Dry
4	10005	Yakulto Oowa Probiotic Drin
5	10006	Oishi Desu Wa Crackers 30g

```
Execution finished without errors.
Result: 5 rows returned in 18ms
At line 1:
SELECT prod_id, prod_name, prod_priceper
FROM Product
WHERE prod_isfastmoving = 1
ORDER BY prod_stockqty ASC;
```

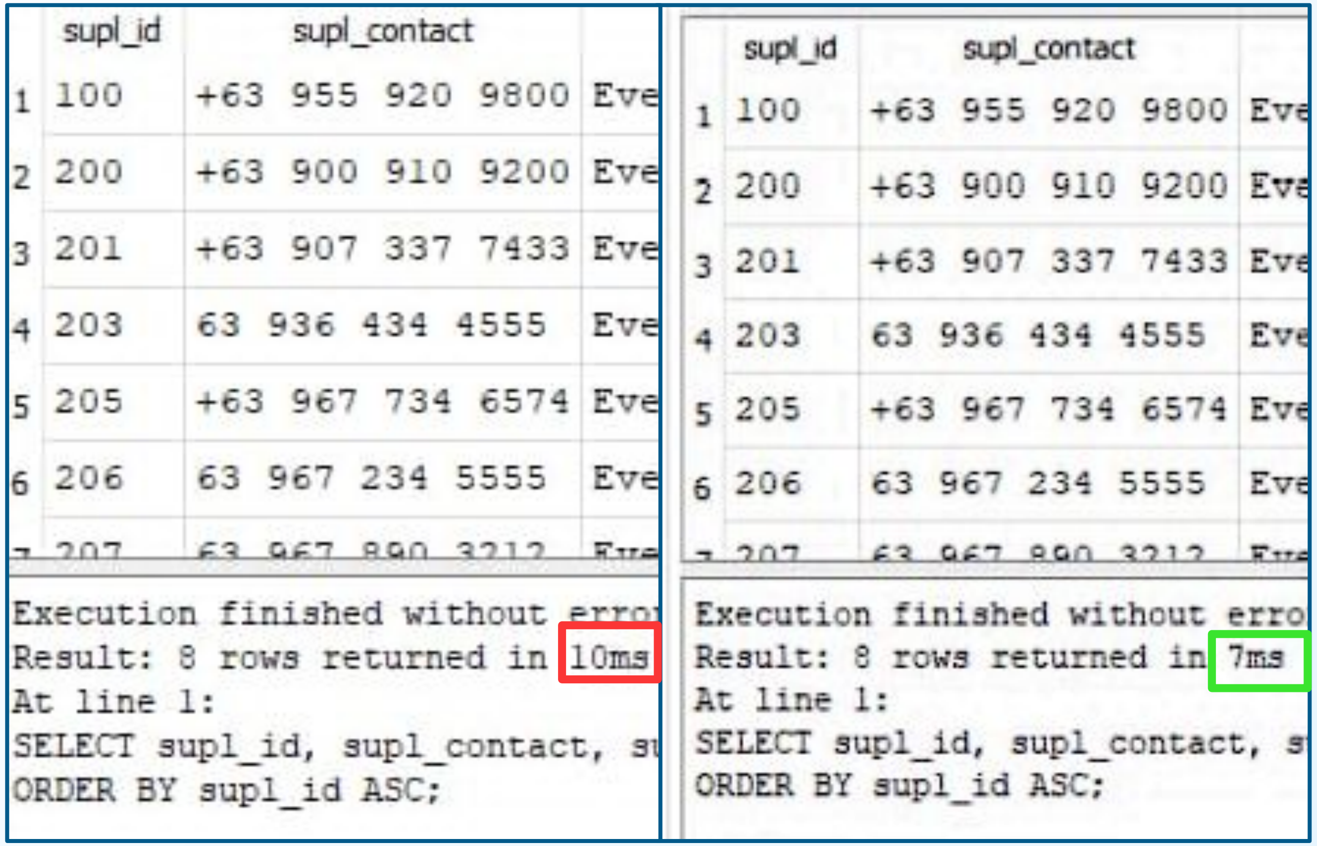
Scenario:

Contrasting stock levels of fast-moving vs. slow-moving products

(22ms vs 18ms)
left right

Optimised Queries & Performance Results

2.
```
SELECT supl_id, supl_contact, supl_schedule FROM
Supplier ORDER BY supl_id ASC;
```



	supl_id	supl_contact	
1	100	+63 955 920 9800	Eve
2	200	+63 900 910 9200	Eve
3	201	+63 907 337 7433	Eve
4	203	63 936 434 4555	Eve
5	205	+63 967 734 6574	Eve
6	206	63 967 234 5555	Eve
7	207	63 967 890 3212	Eve

```
Execution finished without erro
Result: 8 rows returned in 10ms
At line 1:
SELECT supl_id, supl_contact, su
ORDER BY supl_id ASC;
```

	supl_id	supl_contact	
1	100	+63 955 920 9800	Eve
2	200	+63 900 910 9200	Eve
3	201	+63 907 337 7433	Eve
4	203	63 936 434 4555	Eve
5	205	+63 967 734 6574	Eve
6	206	63 967 234 5555	Eve
7	207	63 967 890 3212	Eve

```
Execution finished without erro
Result: 8 rows returned in 7ms
At line 1:
SELECT supl_id, supl_contact, s
ORDER BY supl_id ASC;
```

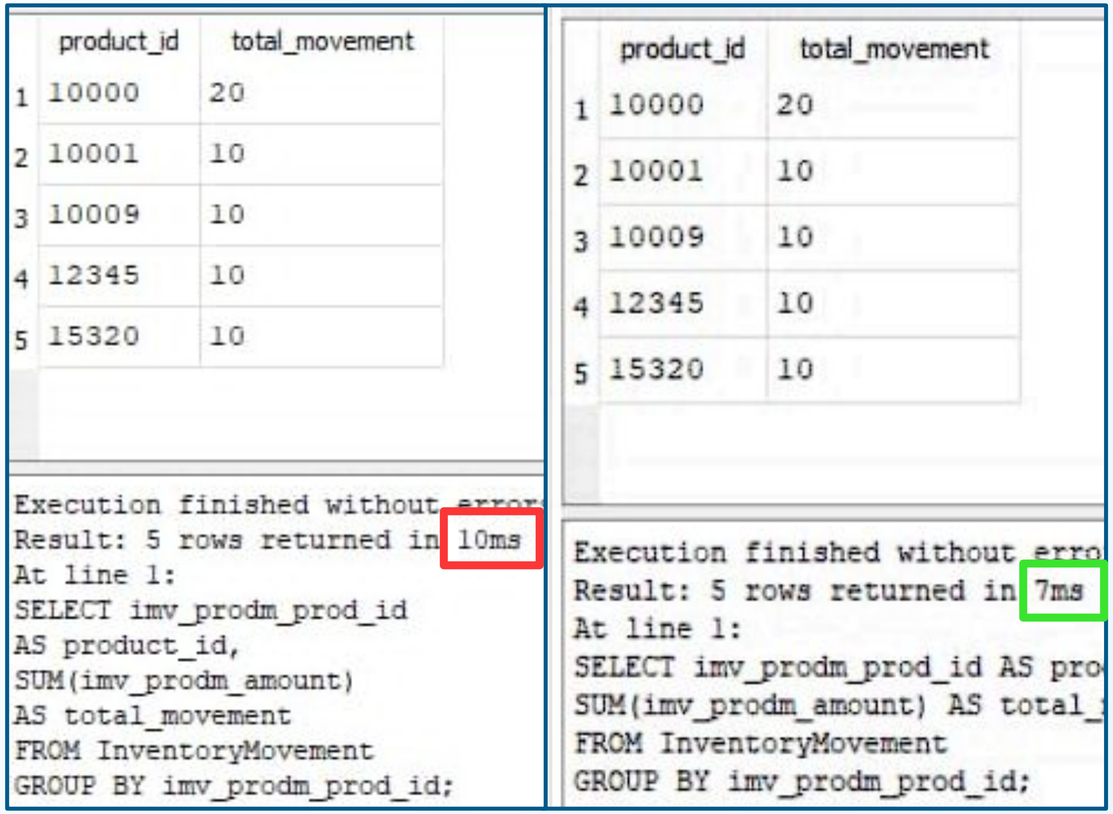
Scenario:

Quick lookup of Supplier contact number given their ID (10ms vs 7ms)

left right

Optimised Queries & Performance Results

3.
```
SELECT imv_prodm_prod_id AS product_id,
SUM(imv_prodm_amount) AS total_movement
FROM InventoryMovement GROUP BY imv_prodm_prod_id;
```



	product_id	total_movement
1	10000	20
2	10001	10
3	10009	10
4	12345	10
5	15320	10

```
Execution finished without error
Result: 5 rows returned in 10ms
At line 1:
SELECT imv_prodm_prod_id
AS product_id,
SUM(imv_prodm_amount)
AS total_movement
FROM InventoryMovement
GROUP BY imv_prodm_prod_id;
```

	product_id	total_movement
1	10000	20
2	10001	10
3	10009	10
4	12345	10
5	15320	10

```
Execution finished without erro
Result: 5 rows returned in 7ms
At line 1:
SELECT imv_prodm_prod_id AS pro
SUM(imv_prodm_amount) AS total_
FROM InventoryMovement
GROUP BY imv_prodm_prod_id;
```

Scenario:

Measure total Inventory Movement of Products

(10ms vs 7ms)
left right

Conclusions & Recommendations

Conclusions: *So what now?*

1. **Model properly.** Keep your friends close and your entities closer.
2. **Check relations.** Study and reflect on how your entities would interact in the real world.
3. **Normalisation today** prevents problems in the future and means you get treats as the data architect. :3
4. **Anticipate frequented columns.** Creating indexes on them promotes performance and scalability.

Recommendations:

What could be next?

1. **Sales & Accounting** system
2. **Customer Relations** system (points, rewards...)
3. **Analysis & Forecasting** System (apply CIT1)

... but that's for the ERP majors to do.

C’est fini.

Merci de votre attention !

Thank you for listening!

RainbowDB

Q&A

Any questions for us?

RainbowDB