**Custom Database Report:**

Name: SM Ragib Rezwan

ID: 103172423

Content List:

* Overview of the database
  + Background of company owning the database
  + Main use of the database
* Simplified UML diagram
* Tables, attributes, data types and the reasoning behind them
  + table Product
  + Table subsidiary\_company\_details
  + Table shipments
  + Table Orders
  + Table Customers\_details
  + Table Employee\_details
  + Table Individual\_Employee\_position
  + Table Employee\_position
  + Table Employee\_and\_Branch\_details
  + Table Branch\_details
* Scripts used to make the database
* Scripts used to insert data for all the tables
* Use of 5 joins in the database:

1) Use Join to make a view with full employee details (including position and their branch names)

2) Use Join to make a view for full product details (including subsidiary company name

3) Use Join to make a view for order stating name of employee and product and customer buying it

4) Use Join to find which product was ordered more than once

5) Use Join to find what the weird customer had ordered in details:

* Reference

**Overview of the database:**

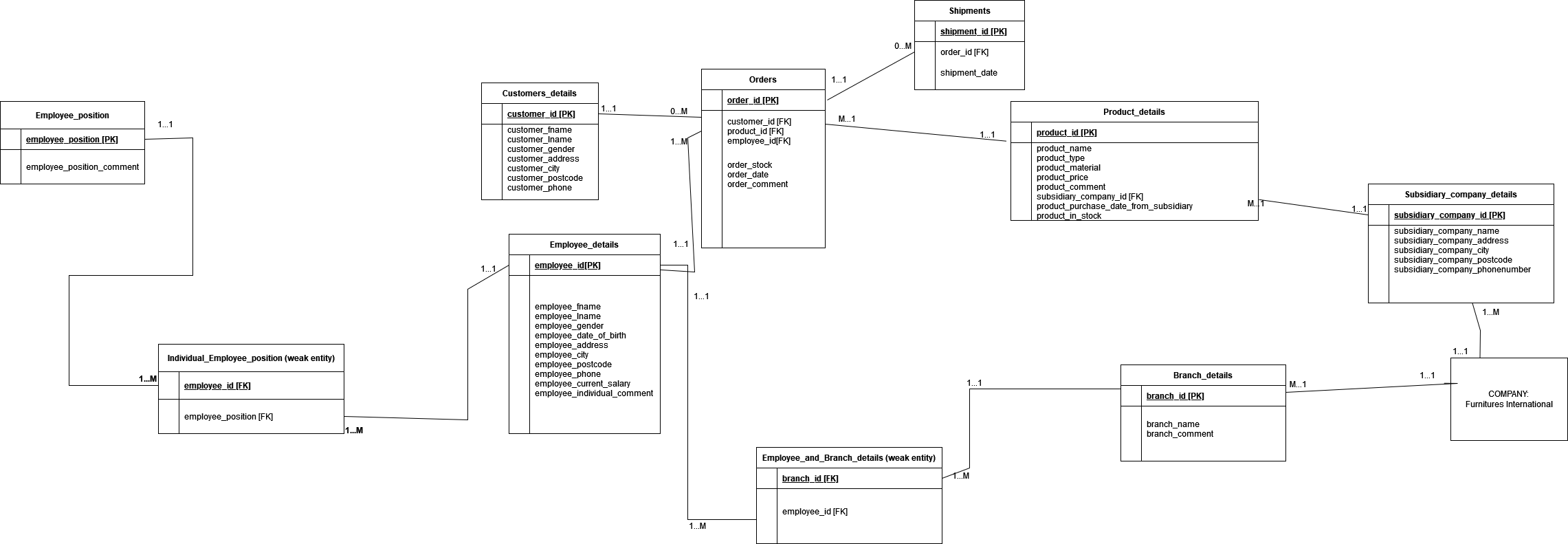
Background of company owning the database:

This is a database made for the Company called Furnitures International. Although it may seem like any other ordinary furniture selling shop, its current situation is rather serious. Previously they had been a big company, controlling a significant portion in the furniture market industry in Southeast Asia with branches in all relevant countries. But unfortunately, they had become a victim of a hostile takeover, losing almost all of their business and employees in the process. Furthermore, they were unable to cope with Covid-19 Crisis which led to further losses in personnel and assets. Now, they are currently trying to build up their business from scratch, trying to make the connections and gather employees using their last 2 branches and few subsidiary companies that they still have connection with.

Main use of the database:

This database will mainly be used to keep track of employee information, Customer information, product information, subsidiary company information, order details and shipping. Thus I used MYSQL here as it is highly structural, provides robust transactions following ACID(atomic, consistent, isolated, durable), etc. Furthermore, data can also be quickly updated and inserted into relevant tables via short queries by using MYSQL.

**Simplified UML Diagram :**



**Tables, attributes, data types and the reasoning behind them:**

1. Table Product

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Datatype | Null or not | Justification |
| product\_id | Varchar(10) | NOT NULL | The Id is a combination of letter and number to kept it varchar to save space and also allow alphanumeric input.  Also all products must have an id so kept it not null |
| product\_name | Varchar(20) | NOT NULL | The name is made of letter so kept it varchar to save space and also allow alphabet input.  Also all products must have a name so kept it not null |
| product\_type | Varchar(8) | NOT NULL | Fixed product type to only let chair, table, sofa, wardrobe as input. So kept it Varchar as letter input and also to save space.  All products have a type so kept it as not null |
| product\_material | Varchar(7) | NOT NULL | Fixed product type to only let wood, plastic, glass as input. So kept it Varchar as letter input and also to save space.  All products are made of a material so kept it as not null |
| product\_price | Int(6) UNISGNED | NOT NULL | Price of products sold by company is in int and wont go above 6 digits, so kept it as int(6). Also it cant be negative so unsigned.  All products have a price, so kept it as not null |
| product\_comment | Varchar(50) | NULL | This is the individual comment about the product so kept it varchar as letter and to save space.  Product may or may not have a comment so kept it as null |
| subsidiary\_company\_id | Varchar(10) | NOT NULL | This is the id of the subsidiary company product was purchased from. So kept it as varchar for alphanumeric input and also to save space.  All products have a subsidiary company id so kept it as not null |
| product\_purchase\_date\_from\_subsidiary | Date | NOT NULL | This is the date product was the last purchased from the subsidiary company. So kept it as date type.  All products have a date when it had been last purchase from subsidiary company |
| Product\_in\_stock | Int (3) | NULL | Kept it int 3 as company currently don’t have storage to store more than 999 product of a type.  Also made it Nullable as some products may be newly introduced and thus they may not have stock for it yet.    (alternatively, it can be kept as not null, but then 0 has to be kept as default for the new product introduction case) |

**Primary key name**: product\_id

**Foreign key names**: subsidiary\_company\_id

1. Table subsidiary\_company\_details

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Datatype | Null or not | Justification |
| Subsidiary\_company\_id | Varchar(10) | NOT NULL | The Id is a combination of letter and number to kept it varchar to save space and also allow alphanumeric input.  Also all company must have an id so kept it not null |
| Subsidiary\_company\_name | Varchar(20) | NOT NULL | The name is made of letter so kept it varchar to save space and also allow alphabet input.  Also all company must have a name so kept it not null |
| Subsidiary\_company\_address | Varchar(40) | NOT NULL | Address is made of letter and numbers, so Varchar as alphanumeric input and also to save space.  All companies have their own addresses and so not null |
| Subsidiary\_company\_city | Varchar(20) | NOT NULL | City is made of letter, so Varchar as letter input and also to save space.  All companies have their own city and so not null |
| Subsidiary\_company\_postcode | Int(4) UNSIGNED | NOT NULL | PostCode is made of numbers, so int as number input and restricted it to 4 digits as postcode only have 4 digits.  All companies have their own postcode and so not null |
| Subsidiary\_company\_phonenumber | Bigint(13) UNISGNED | NOT NULL | Phonenumber is made of numbers, so made it bigint as number input and 13 digits. Didn’t use int as it can only take from 0 to 4294967295 when unsigned which is not enough!  All companies have their own number and so not null |

**Primary key name**: subsidiary\_company\_id

1. Table shipments

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Datatype | Null or not | Justification |
| shipment\_id | Varchar(10) | NOT NULL | The Id is a combination of letter and number to kept it varchar to save space and also allow alphanumeric input.  Also all shipment must have an id so kept it not null |
| Order\_id | Varchar(10) | NOT NULL | The Id is a combination of letter and number to kept it varchar to save space and also allow alphanumeric input.  Also all shipment must have an order\_id to correspond to it, so kept it not null |
| Shipment\_date | Date | NOT NULL | Ship date is bsacially a date so kept it as date type.  All shipments have their own shipping date to kept it not null |

**Primary key name**: shipment\_id

**Foreign key names**: order\_id

1. Table Orders

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Datatype | Null or not | Justification |
| order\_id | Varchar(10) | NOT NULL | The Id is a combination of letter and number to kept it varchar to save space and also allow alphanumeric input.  Also all orders must have an order id so kept it not null |
| customer\_id | Varchar(10) | NOT NULL | The Id is a combination of letter and number to kept it varchar to save space and also allow alphanumeric input.  Also all orders must have an customer id so kept it not null |
| product\_id | Varchar(10) | NOT NULL | The Id is a combination of letter and number to kept it varchar to save space and also allow alphanumeric input.  Also all orders must have a product id so kept it not null |
| employee\_id | Varchar(10) | NOT NULL | The Id is a combination of letter and number to kept it varchar to save space and also allow alphanumeric input.  Also all orders must have an employeer id so kept it not null |
| order\_stock | Int(3) | NOT NULL | Order Stock is basically number of products purchased to kept it int to allow number input. Also it it wont be above 3 digit as originally stock stored is not more than 3 digits  Also all orders must have order\_stock so not null |
| order\_date | date | NOT NULL | Order date is basically the date product has been ordered. So kept it date type.  Also all orders must have an order date to kept it as not null |
| Order\_comment | Varchar(50) | NULL | Comment is combination of number and letters to kept it as varhar to save space and also allow alphanumeric input.  An order may or may not have any comment so it can be null |

**Primary key name:** order\_id

**Foreign key names:** customer\_id, product\_id, employee\_id

**Table type:** It is a weak entity table

1. Table Customers\_details

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Datatype | Null or not | Justification |
| customer\_id | Varchar(10) | NOT NULL | The Id is a combination of letter and number to kept it varchar to save space and also allow alphanumeric input.  Also all customers must have an customer id so kept it not null |
| customer\_fname | Varchar(20) | NOT NULL | The fname is basically the firstname of the customer so kept it as varchar to save space and also allow letter input.  Also all customers must have firstname so kept it not null |
| customer\_lname | Varchar(20) | NOT NULL | The lname is basically the lastname of the customer so kept it as varchar to save space and also allow letter input.  Also all customers must have lastname so kept it not null |
| customer\_gender | Varchar(1) | NOT NULL | Gender basically has 3 possiblities: Male(M), Female (F), Unknown(U). So restricted its input to M,F,U  Also all customers must have gender so kept it not null |
| customer\_address | Varchar(20) | NOT NULL | The address is basically the address of the customer so kept it as varchar to save space and also allows letter input.  Also all customers must have address so kept it not null |
| customer\_city | Varchar(20) | NOT NULL | The city is basically the city of the customer so kept it as varchar to save space and also allows letter input.  Also all customers must have city so kept it not null |
| customer\_postcode | Int(4) | NOT NULL | The postis basically the postcode of the customer living area. So made it int and restricted input to 4 digits.  Also all customers must have postcode so kept it not null |
| customer\_phone | Bigint(13) | NOT NULL | Phonenumber is made of numbers, so made it bigint as number input and 13 digits. Didn’t use int as it can only take from 0 to 4294967295 when unsigned which is not enough!  All customers have their own number and so not null |

**Primary key name:** customer\_id

1. Table Employee\_details

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Datatype | Null or not | Justification |
| employee\_id | Varchar(10) | NOT NULL | The Id is a combination of letter and number to kept it varchar to save space and also allow alphanumeric input.  Also all employee must have an customer id so kept it not null |
| employee \_fname | Varchar(20) | NOT NULL | The fname is basically the firstname of the employee so kept it as varchar to save space and also allow letter input.  Also all employee must have firstname so kept it not null |
| employee \_lname | Varchar(20) | NOT NULL | The lname is basically the lastname of the employee so kept it as varchar to save space and also allow letter input.  Also all employee must have lastname so kept it not null |
| employee \_gender | Varchar(1) | NOT NULL | Gender basically has 3 possiblities: Male(M), Female (F), Unknown(U). So restricted its input to M,F,U  Also all employee must have gender so kept it not null |
| Employee\_date\_of\_birth | date | NOT NULL | The date of birth is basically the birthday of the employee so kept it as date type.  Also all employee must have birthday so kept it not null |
| employee \_address | Varchar(20) | NOT NULL | The address is basically the address of the employee so kept it as varchar to save space and also allows letter input.  Also all employee must have address so kept it not null |
| employee \_city | Varchar(20) | NOT NULL | The city is basically the city of the employee so kept it as varchar to save space and also allows letter input.  Also all employee must have city so kept it not null |
| employee \_postcode | Int(4) | NOT NULL | The postcode is basically the postcode of the employee’s living area. So made it int and restricted input to 4 digits.  Also all employee must have postcode so kept it not null |
| employee \_phone | Bigint(13) | NOT NULL | Phonenumber is made of numbers, so made it bigint as number input and 13 digits. Didn’t use int as it can only take from 0 to 4294967295 when unsigned which is not enough!  All employee have their own number and so not null |
| employee\_current\_salary | Int(10) | NULL | Salary basically is the amount of money employee currently earns per year  Employee may have newly joined the company and thus he may not have received his salary yet and thus kept it null  (alternatively, it is also possible to keep it as not null, but then we will need to keep default as 0) |
| employee\_individual\_comment | Varchar(50) | NULL | This is the individual comment about the employee so kept it varchar as letter and to save space.  employee may or may not have a comment so kept it as null |

**Primary key name:** employee\_id

1. Table Individual\_Employee\_position

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Datatype | Null or not | Justification |
| employee\_id | Varchar(10) | NOT NULL | The Id is a combination of letter and number to kept it varchar to save space and also allow alphanumeric input.  Also all employee must have an customer id so kept it not null |
| employee \_position | Varchar(10) | NOT NULL | Position basically has 3 possiblities:  Manager, technician, normal.  So restricted its input to these and used vachar for letter input and also to save space  Also all employee must have position so kept it not null |

**Foreign key names:** employee\_id, employee\_position

**Table type:** It is a weak entity table

1. Table Employee\_position

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Datatype | Null or not | Justification |
| employee \_position | Varchar(10) | NOT NULL | Position basically has 3 possiblities:  Manager, technician, normal.  So restricted its input to these and used vachar for letter input and also to save space  Also all employee must have position so kept it not null |
| Employee\_position\_comment | Varchar(50) | NOT NULL | This tell details of what the employee at this position does. So kept it as varchar as alphanumeric input and also to save space.  Also made this specific comment not null as tells details about the employee’s position’s responsibility  And thus must have relevant information in it regarding all possible employee positions. |

**Primary key names:** employee\_position

1. Table Employee\_and\_Branch\_details

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Datatype | Null or not | Justification |
| Branch\_id | Varchar(10) | NOT NULL | The Id is a combination of letter and number to kept it varchar to save space and also allow alphanumeric input.  Also all branches must have a branch id so kept it not null |
| Employee\_id | Varchar(10) | NOT NULL | The Id is a combination of letter and number to kept it varchar to save space and also allow alphanumeric input.  Also all employes must have a employee id so kept it not null |

**Foreign key names:** branch\_id, employee\_id

**Table type:** It is a weak entity table

1. Table Branch\_details

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute name | Datatype | Null or not | Justification |
| Branch\_id | Varchar(10) | NOT NULL | The Id is a combination of letter and number to kept it varchar to save space and also allow alphanumeric input.  Also all branches must have a branch id so kept it not null |
| Branch\_name | Varchar(20) | NOT NULL | The name is basically name of branch so kept it varchar to save space and also allow letter input  Also all branches must have a branch name so kept it not null |
| Branch\_comment | Varchar(20) | NULL | This is the individual comment about the branch so kept it varchar as letter and to save space.  branch may or may not have a comment so kept it as null |

**Primary key names:** branch\_id

**Scripts used to make the database:**

CREATE database CompanyNew;

Use CompanyNew;

Create table Branch\_details

(

branch\_id varchar(10) NOT NULL,

branch\_name varchar(20) NOT NULL,

branch\_comment varchar(50),

PRIMARY KEY(branch\_id)

);

Create table Employee\_details

(

employee\_id varchar(10) NOT NULL ,

employee\_fname varchar(20) NOT NULL,

employee\_lname varchar(20) NOT NULL,

employee\_gender varchar(1) NOT NULL

CHECK (employee\_gender IN ('M','F', 'U')),

employee\_date\_of\_birth date NOT NULL,

employee\_address varchar(20) NOT NULL,

employee\_city varchar(20) NOT NULL,

employee\_postcode int(4) UNSIGNED NOT NULL,

employee\_phone BIGINT(13) UNSIGNED NOT NULL,

employee\_current\_salary int(10) UNSIGNED,

employee\_individual\_comment varchar(50),

PRIMARY KEY(employee\_id)

);

Create table Employee\_and\_Branch\_details

(

branch\_id varchar(10) NOT NULL,

employee\_id varchar(10) NOT NULL ,

PRIMARY KEY(branch\_id , employee\_id),

FOREIGN KEY(branch\_id) REFERENCES Branch\_details (branch\_id),

FOREIGN KEY(employee\_id) REFERENCES Employee\_details (employee\_id)

);

Create table Employee\_position

(

employee\_position varchar(10) NOT NULL

CHECK (employee\_position IN ('Manager','Technician', 'Normal')),

employee\_position\_comment varchar(50) NOT NULL,

PRIMARY KEY(employee\_position)

);

Create table Individual\_Employee\_position

(

employee\_id varchar(10) NOT NULL ,

employee\_position varchar(10) NOT NULL

CHECK (employee\_position IN ('Manager','Technician', 'Normal')),

PRIMARY KEY(employee\_id, employee\_position),

FOREIGN KEY(employee\_id) REFERENCES Employee\_details (employee\_id),

FOREIGN KEY(employee\_position) REFERENCES Employee\_position (employee\_position)

);

Create table Subsidiary\_company\_details

(

subsidiary\_company\_id varchar(10) NOT NULL,

subsidiary\_company\_name varchar(20) NOT NULL,

subsidiary\_company\_address varchar(40) NOT NULL,

subsidiary\_company\_city varchar(20) NOT NULL,

subsidiary\_company\_postcode int(4) UNSIGNED NOT NULL,

subsidiary\_company\_phonenumber bigint(13) UNSIGNED NOT NULL,

PRIMARY KEY(subsidiary\_company\_id)

);

Create table Product\_details

(

product\_id varchar(10) NOT NULL ,

product\_name varchar(20) NOT NULL,

product\_type varchar(8) NOT NULL

CHECK (product\_type IN ('chair', 'table', 'sofa', 'wardrobe')),

product\_material varchar(7) NOT NULL

CHECK (product\_material IN ('wood', 'plastic', 'glass')),

product\_price int(6) UNSIGNED NOT NULL,

product\_comment varchar(50),

subsidiary\_company\_id varchar(10) NOT NULL,

product\_purchase\_date\_from\_subsidiary Date NOT NULL,

product\_in\_stock int(3) UNSIGNED NOT NULL,

PRIMARY KEY(product\_id ,subsidiary\_company\_id),

FOREIGN KEY(subsidiary\_company\_id) REFERENCES Subsidiary\_company\_details (subsidiary\_company\_id)

);

Create table Customers\_details

(

customer\_id varchar(10) NOT NULL,

customer\_fname varchar(20) NOT NULL,

customer\_lname varchar(20) NOT NULL,

customer\_gender varchar(1) NOT NULL

CHECK (customer\_gender IN ('M','F', 'U')),

customer\_address varchar(20) NOT NULL,

customer\_city varchar(20) NOT NULL,

customer\_postcode int(4) UNSIGNED NOT NULL,

customer\_phone BIGINT(13) UNSIGNED NOT NULL,

PRIMARY KEY(customer\_id)

);

Create table Orders

(

order\_id varchar(10) NOT NULL,

customer\_id varchar (10) NOT NULL,

product\_id varchar(10) NOT NULL,

employee\_id varchar(10) NOT NULL ,

order\_stock int(3) NOT NULL,

order\_date date NOT NULL,

order\_comment varchar(50),

PRIMARY KEY(order\_id , customer\_id ,product\_id, employee\_id),

FOREIGN KEY(customer\_id) REFERENCES Customers\_details (customer\_id),

FOREIGN KEY(product\_id) REFERENCES Product\_details (product\_id),

FOREIGN KEY( employee\_id) REFERENCES Employee\_details ( employee\_id)

);

Create table Shipments

(

shipment\_id varchar(10) NOT NULL,

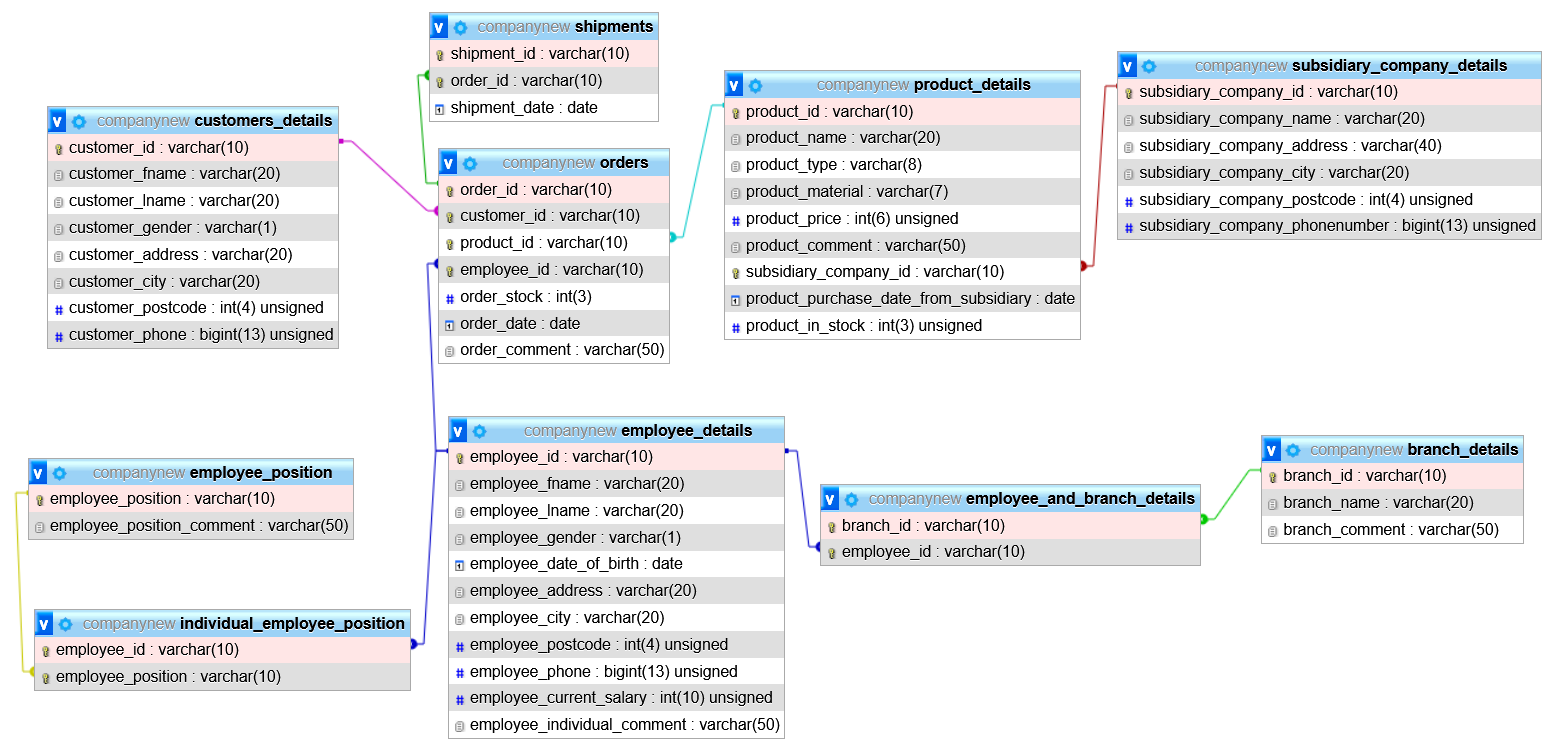
order\_id varchar(10) NOT NULL,

shipment\_date date NOT NULL,

PRIMARY KEY(shipment\_id ,order\_id),

FOREIGN KEY(order\_id) REFERENCES Orders (order\_id)

);



**Scripts used to insert data for all the tables:**

INSERT INTO customers\_details

(customer\_id, customer\_fname, customer\_lname, customer\_gender,

customer\_address, customer\_city, customer\_postcode, customer\_phone)

VALUES ('HABCA00024', 'Sultan', 'Ahmed', 'M',

'Road 2, Gulshan', 'Dhaka', 2141, 8801792458030);

INSERT INTO customers\_details

(customer\_id, customer\_fname, customer\_lname, customer\_gender,

customer\_address, customer\_city, customer\_postcode, customer\_phone)

VALUES ('ADBCA00554', 'Kawsar', 'Hossain', 'M',

'Road 10, Bonani', 'Dhaka', 4156, 8801796958971);

INSERT INTO customers\_details

(customer\_id, customer\_fname, customer\_lname, customer\_gender,

customer\_address, customer\_city, customer\_postcode, customer\_phone)

VALUES ('FABCA09548', 'Robert', 'Brown', 'M',

'Road 10, Modina', 'Sylhet', 3114, 8801892884308);

INSERT INTO customers\_details

(customer\_id, customer\_fname, customer\_lname, customer\_gender,

customer\_address, customer\_city, customer\_postcode, customer\_phone)

VALUES ('ALBCA11020', 'Anne', 'Stuward', 'F',

'Road 2, Rampura', 'Dhaka', 6915, 8801692433190);

INSERT INTO customers\_details

(customer\_id, customer\_fname, customer\_lname, customer\_gender,

customer\_address, customer\_city, customer\_postcode, customer\_phone)

VALUES ('GABCA07324', 'Salma', 'Begum', 'F',

'Road 21, Gulshan', 'Dhaka', 2114, 8801592457044);

INSERT INTO subsidiary\_company\_details

(subsidiary\_company\_id, subsidiary\_company\_name, subsidiary\_company\_address, subsidiary\_company\_city,

subsidiary\_company\_postcode, subsidiary\_company\_phonenumber)

VALUES ('SCOMP00001', 'Office Products', 'Road 49, Gulshan', 'Dhaka',

2441, 8801972448031);

INSERT INTO subsidiary\_company\_details

(subsidiary\_company\_id, subsidiary\_company\_name, subsidiary\_company\_address, subsidiary\_company\_city,

subsidiary\_company\_postcode, subsidiary\_company\_phonenumber)

VALUES ('SCOMP00444', 'Stylish Products', 'Road 2, Gulshan', 'Dhaka',

2411, 8801973448031);

INSERT INTO subsidiary\_company\_details

(subsidiary\_company\_id, subsidiary\_company\_name, subsidiary\_company\_address, subsidiary\_company\_city,

subsidiary\_company\_postcode, subsidiary\_company\_phonenumber)

VALUES ('SCOMP00051', 'RFL', 'Road 4,Bonani', 'Dhaka',

6518, 8801745336711);

INSERT INTO subsidiary\_company\_details

(subsidiary\_company\_id, subsidiary\_company\_name, subsidiary\_company\_address, subsidiary\_company\_city,

subsidiary\_company\_postcode, subsidiary\_company\_phonenumber)

VALUES ('SCOMP05001', 'Glassify', 'Road 4, Kumarpara', 'Sylhet',

3141, 8801745336700);

INSERT INTO product\_details

(product\_id, product\_name, product\_type, product\_material,

product\_price, product\_comment, subsidiary\_company\_id, product\_purchase\_date\_from\_subsidiary,product\_in\_stock)

VALUES ('PRDTA00103', 'Office chair', 'chair', 'wood',

4000, 'good office chair made of wood', 'SCOMP00001', '2020-12-12',70);

INSERT INTO product\_details

(product\_id, product\_name, product\_type, product\_material,

product\_price, product\_comment, subsidiary\_company\_id, product\_purchase\_date\_from\_subsidiary, product\_in\_stock)

VALUES ('PHGTA00144', 'Office table', 'table', 'wood',

6000, 'good office table made of wood', 'SCOMP00001', '2019-2-16', 40);

INSERT INTO product\_details

(product\_id, product\_name, product\_type, product\_material,

product\_price, product\_comment, subsidiary\_company\_id, product\_purchase\_date\_from\_subsidiary, product\_in\_stock)

VALUES ('PRDTG00210', 'stylish chair', 'chair', 'wood',

10000, 'good chair made of wood', 'SCOMP00444', '2020-12-15', 50);

INSERT INTO product\_details

(product\_id, product\_name, product\_type, product\_material,

product\_price, product\_comment, subsidiary\_company\_id, product\_purchase\_date\_from\_subsidiary, product\_in\_stock)

VALUES ('PRDTA00140', 'RFL sofa', 'sofa', 'plastic',

500, 'good RFL chair made of plastic', 'SCOMP00051', '2018-10-12', 90);

INSERT INTO product\_details

(product\_id, product\_name, product\_type, product\_material,

product\_price, product\_comment, subsidiary\_company\_id, product\_purchase\_date\_from\_subsidiary, product\_in\_stock)

VALUES ('PROOA00103', 'clear wardrobe', 'wardrobe', 'glass',

8000, 'good wardrobe made of glass', 'SCOMP05001', '2020-10-10', 90);

INSERT INTO employee\_details

(employee\_id, employee\_fname, employee\_lname, employee\_gender, employee\_date\_of\_birth,

employee\_address, employee\_city, employee\_postcode, employee\_phone, employee\_current\_salary, employee\_individual\_comment)

VALUES ('EMPLY44556', 'Karim', 'Kombol', 'M',

'1990-10-24', 'Road 45, Gulshan', 'Dhaka', 2356, 8801731246924, 80000, 'good old employee');

INSERT INTO employee\_details

(employee\_id, employee\_fname, employee\_lname, employee\_gender, employee\_date\_of\_birth,

employee\_address, employee\_city, employee\_postcode, employee\_phone, employee\_current\_salary, employee\_individual\_comment)

VALUES ('EMPLE24598', 'Bokul', 'Ahmed', 'F',

'1995-1-2', 'Road 40, Gulshan', 'Dhaka', 2557, 8801933246924, 60000, 'clumsy employee');

INSERT INTO employee\_details

(employee\_id, employee\_fname, employee\_lname, employee\_gender, employee\_date\_of\_birth,

employee\_address, employee\_city, employee\_postcode, employee\_phone, employee\_current\_salary, employee\_individual\_comment)

VALUES ('ETYLE24594', 'Mokles', 'Miah', 'M',

'2000-10-20', 'Road 45, Subidbazar', 'Sylhet', 3156, 8801731296924, 40000, 'Lazy employee');

INSERT INTO employee\_details

(employee\_id, employee\_fname, employee\_lname, employee\_gender, employee\_date\_of\_birth,

employee\_address, employee\_city, employee\_postcode, employee\_phone, employee\_current\_salary, employee\_individual\_comment)

VALUES ('QWRTY11111', 'Raima', 'Karim', 'F',

'2001-3-4', 'Road 4, Modinamarket', 'Sylhet', 4156, 8801455672298, 80000, 'Diligent employee');

INSERT INTO employee\_details

(employee\_id, employee\_fname, employee\_lname, employee\_gender, employee\_date\_of\_birth,

employee\_address, employee\_city, employee\_postcode, employee\_phone, employee\_current\_salary, employee\_individual\_comment)

VALUES ('QSART45692', 'Kashem', 'Uddin', 'M',

'1995-5-5', 'Road 8, Rampura', 'Dhaka', 2400, 8801459272298, 90000, 'old techy employee');

INSERT INTO branch\_details

(branch\_id, branch\_name, branch\_comment)

VALUES ('BRNCH40002', 'Dhaka Branch', 'Exclusive Branch in Dhaka');

INSERT INTO branch\_details

(branch\_id, branch\_name, branch\_comment)

VALUES ('BRNCH20003', 'Sylhet Branch', 'Exclusive Branch in Sylhet');

INSERT INTO employee\_and\_branch\_details

(branch\_id, employee\_id)

VALUES ('BRNCH40002', 'EMPLY44556');

INSERT INTO employee\_and\_branch\_details

(branch\_id, employee\_id)

VALUES ('BRNCH40002', 'EMPLE24598');

INSERT INTO employee\_and\_branch\_details

(branch\_id, employee\_id)

VALUES ('BRNCH20003', 'ETYLE24594');

INSERT INTO employee\_and\_branch\_details

(branch\_id, employee\_id)

VALUES ('BRNCH20003', 'QWRTY11111');

INSERT INTO employee\_and\_branch\_details

(branch\_id, employee\_id)

VALUES ('BRNCH40002', 'QSART45692');

INSERT INTO employee\_position

(employee\_position, employee\_position\_comment)

VALUES ('Manager', 'responsible for controlling group of staff');

INSERT INTO employee\_position

(employee\_position, employee\_position\_comment)

VALUES ('Technician', 'maintains technical equipment');

INSERT INTO employee\_position

(employee\_position, employee\_position\_comment)

VALUES ('Normal', 'ordinary person employeed');

INSERT INTO individual\_employee\_position

(employee\_id, employee\_position)

VALUES ('EMPLY44556', 'Manager');

INSERT INTO individual\_employee\_position

(employee\_id, employee\_position)

VALUES ('EMPLE24598', 'Normal');

INSERT INTO individual\_employee\_position

(employee\_id, employee\_position)

VALUES ('ETYLE24594', 'Normal');

INSERT INTO individual\_employee\_position

(employee\_id, employee\_position)

VALUES ('QWRTY11111', 'Manager');

INSERT INTO individual\_employee\_position

(employee\_id, employee\_position)

VALUES ('QSART45692', 'Technician');

INSERT INTO orders

(order\_id, customer\_id, product\_id, employee\_id, order\_stock, order\_date, order\_comment)

VALUES ('ORDRE45679', 'HABCA00024', 'PHGTA00144', 'EMPLE24598',1, '2021-5-24', 'customer felt weird');

INSERT INTO orders

(order\_id, customer\_id, product\_id, employee\_id, order\_stock, order\_date, order\_comment)

VALUES ('ORDRE45680', 'HABCA00024', 'PRDTA00103', 'EMPLE24598',1, '2021-5-24', 'customer felt weird');

INSERT INTO orders

(order\_id, customer\_id, product\_id, employee\_id, order\_stock, order\_date, order\_comment)

VALUES ('ORDRE44990', 'FABCA09548', 'PROOA00103', 'ETYLE24594',10, '2021-6-26', 'good old customer back');

INSERT INTO orders

(order\_id, customer\_id, product\_id, employee\_id, order\_stock, order\_date, order\_comment)

VALUES ('ORDRE49100', 'GABCA07324', 'PRDTA00103', 'EMPLE24598',10, '2021-7-7', 'customer wants to track this');

INSERT INTO shipments

(shipment\_id, order\_id, shipment\_date)

VALUES ('SHPIN30078', 'ORDRE45679', '2021-5-26');

INSERT INTO shipments

(shipment\_id, order\_id, shipment\_date)

VALUES ('SHPIN30079', 'ORDRE45680', '2021-5-26');

INSERT INTO shipments

(shipment\_id, order\_id, shipment\_date)

VALUES ('SHPIN40705', 'ORDRE44990', '2021-6-28');

INSERT INTO shipments

(shipment\_id, order\_id, shipment\_date)

VALUES ('SHPIN55555', 'ORDRE49100', '2021-7-9');

**Use of 5 joins:**

Some of the joins I used in the database and their purposes: (added reasoning for the places where I made views and sides notes for normal joins)

1. Use Join to make a view with full employee details (including position and their branch names)

Reason:

To make it easier for employer to find all employee details in a single view instead of looking at several tables.

Also, since view is a stored query, it is already pre-optimized and thus would execute faster when opened next time as no need to re-optimize it (As query execution strategy has already been decided on by DBMS when the view was first created). Also, no need to use join and query through multiple tables again and again to find the data. Thus, overall it increases the performance of the database.

Commands:

Create View Full\_Employee\_Details as

Select employee\_id, employee\_fname, employee\_lname, employee\_gender, employee\_date\_of\_birth, employee\_address,

employee\_city, employee\_postcode, employee\_phone, employee\_current\_salary, employee\_position,branch\_name, employee\_individual\_comment

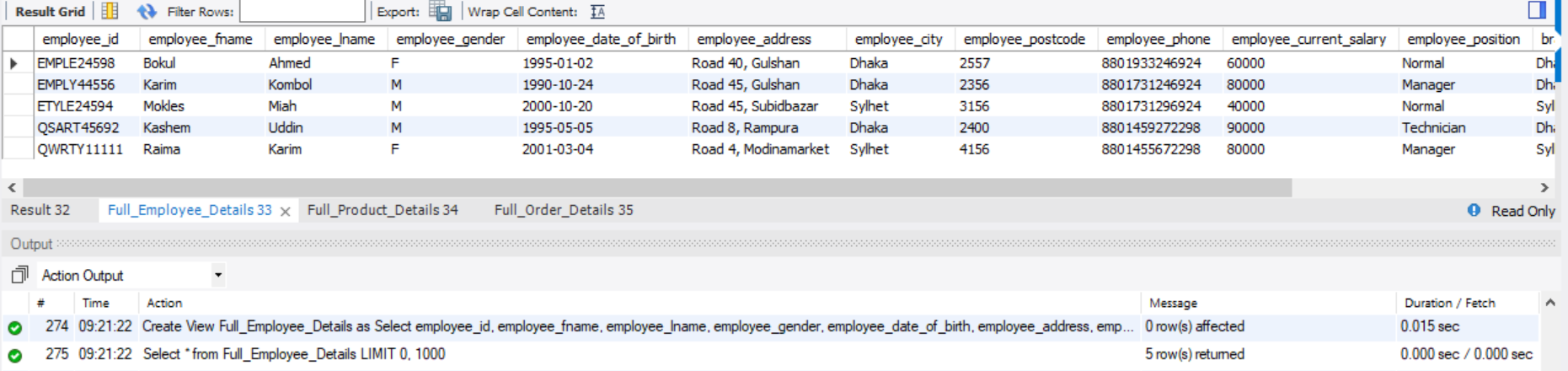
From employee\_details e Natural Join employee\_and\_branch\_details eb Natural Join branch\_details b NATURAL Join individual\_employee\_position ie NATURAL Join employee\_position

Order by employee\_id;

Select \*

from Full\_Employee\_Details;

Output:



1. Use Join to make a view for full product details (including subsidiary company name

Reason:

To have a single view with both poduct, subsidiary company information. This makes it easier for company to restock the products quickly, as they can easily see which company its from and the phone no of that company. Also there is company’s city included and thus they can use it to estimate how long it will take for the product to be delivered to them.

Also, since view is a stored query, it is already pre-optimized and thus would execute faster when opened next time as no need to re-optimize it (As query execution strategy has already been decided on by DBMS when the view was first created). Also, no need to use join and query through multiple tables again and again to find the data. Thus, overall it increases the performance of the database.

Command:

Create View Full\_Product\_Details as

Select product\_id, product\_name, product\_type, product\_material, product\_price, product\_comment, subsidiary\_company\_id,

subsidiary\_company\_name, subsidiary\_company\_city, subsidiary\_company\_phonenumber, product\_purchase\_date\_from\_subsidiary,

product\_in\_stock

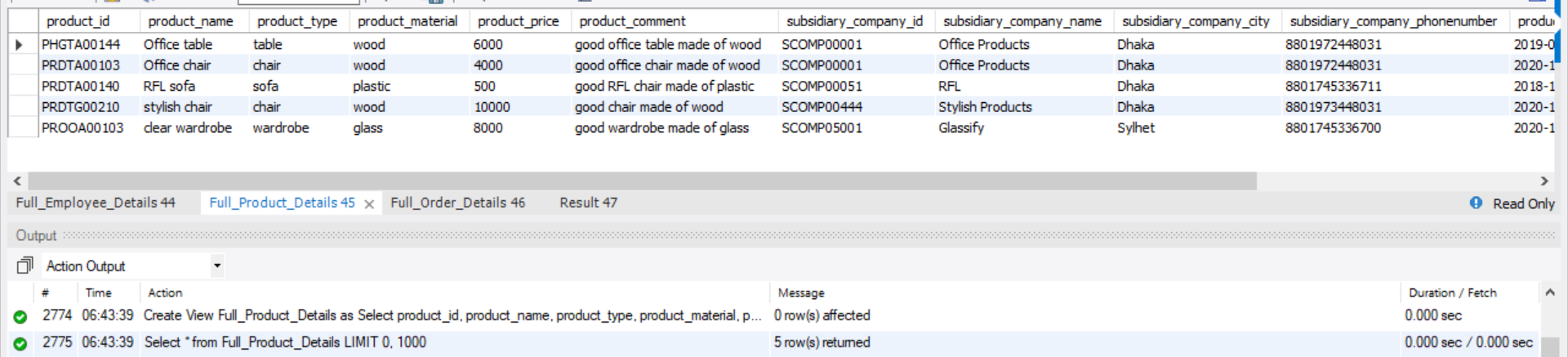
From product\_details pd Natural Join subsidiary\_company\_details scd

Order by product\_id;

Select \*

from Full\_Product\_Details;

Output:



1. Use Join to make a view for order stating name of employee and product and customer buying it

Reason:

To have a single view to see all order details including customer, employee who sold it, total cost of order, etc.

Also, since view is a stored query, it is already pre-optimized and thus would execute faster when opened next time as no need to re-optimize it (As query execution strategy has already been decided on by DBMS when the view was first created). Also, no need to use join and query through multiple tables again and again to find the data. Thus, overall it increases the performance of the database.

Command:

Create View Full\_Order\_Details as

Select order\_id, customer\_id, customer\_fname, customer\_lname, customer\_gender, product\_id, product\_name, (SUM(product\_price\*order\_stock)) as Order\_COST,employee\_id,

employee\_fname, employee\_lname, order\_stock, order\_date, order\_comment

From orders o Natural Join product\_details p Natural Join customers\_details c Natural Join employee\_details e

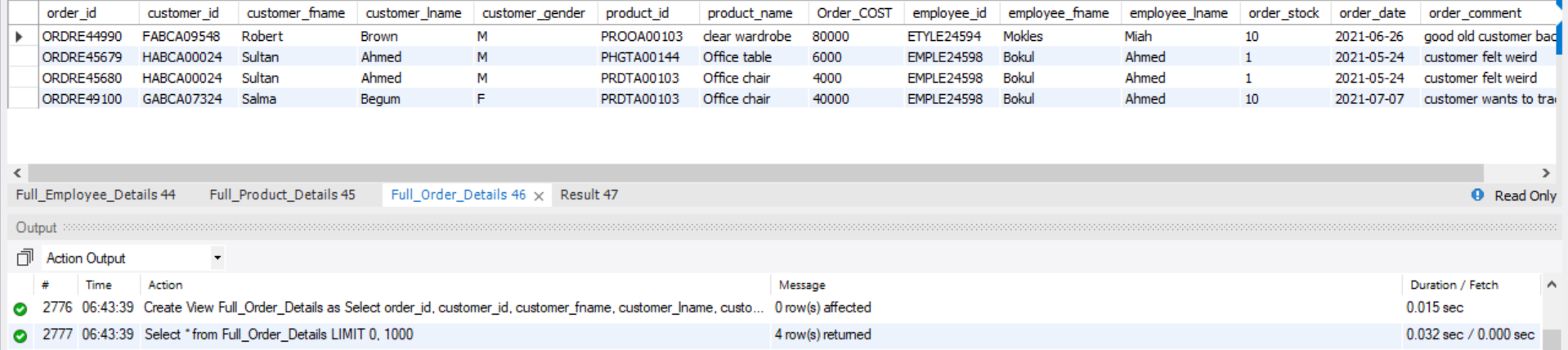
Group by order\_id

Order by order\_id;

Select \*

from Full\_Order\_Details;

Output



1. Use join to find which product was ordered more than once:

Commands:

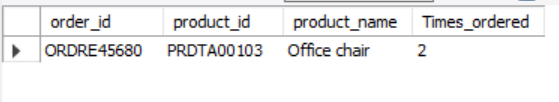
Select order\_id, product\_id, product\_name, (Count(product\_id)) as Times\_ordered

From orders o Natural Join product\_details p

group by product\_id

Having Times\_ordered>1;

Output:



Sidenote:

(We can also directly query from the “view” instead of using the joins, if that has been made beforehand. That will be more efficient and better as joins are expensive as it needs to look though many tables and compare datas before querying)

1. Use Join to find what the weird customer had ordered in details:

Command:

Select order\_id, customer\_id, customer\_fname, customer\_lname, product\_id, product\_name, (SUM(product\_price\*order\_stock)) as Order\_COST,

order\_stock, order\_date, order\_comment

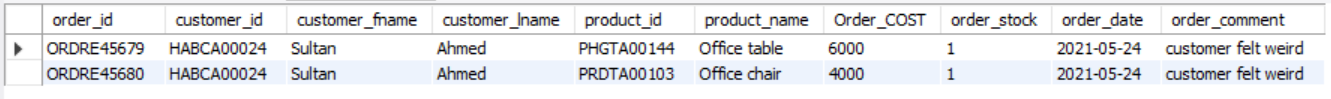
From orders o Natural Join product\_details p Natural Join customers\_details c

Group by order\_id

Having customer\_id = 'HABCA00024'

Order by order\_id;

Output:



Sidenote:

(We can also directly query from the “view” instead of using the joins, if that has been made beforehand. That will be more efficient and better as joins are expensive as it needs to look though many tables and compare datas before querying)

**Reference:**

* Information from COS20015 (Fundamentals of database management) slide notes regarding MYSQL