Retail Store Management

—

Course Title: Advanced Database (CMP7214)

Course Leader: Dr Parnia Samimi

Project By:

Aditya Batra (21144695)

Darshan Tamboli (21171488)

Index

1. Domain Inception: 2
2. Database Analysis: 2
   - 2.1. Business Situation: 2
   - 2.2. Business Rules: 2
   - 2.3. List of Entity/Attributes: 3
   - 2.4. Simple Relationships: 3
   - 2.5. Connectivity, Cardinalities and Participation: 4
   - 2.6. ERD Mapping: 6
     - 2.6.1. Mapping 1: 1 Relationships: 6
     - 2.6.2. Mapping 1: N Relationships: 7
     - 2.6.3. Mapping M: N Relationships: 9
3. Database Design: 10
4. Normalization: 11
5. Database Implementation: 13
   - 5.1. Table Creation: 13
   - 5.2. Getting Data into Table: 16
   - 5.3. SQL Queries testing: 23
6. Conclusions: 27

1. Project Inception:

Retail shops help people to fulfil the daily needs of groceries. Tracking such a huge amount of data related to products, customers, employees, orders, payments, and suppliers is essential for retail shops. Managing all these records on paper is untrustworthy, susceptible and sloppy. The motto of our project is to create a rational database to access, manage, modify, and update data as per the requirement.

2. Database Analysis:

We have worked on a database to support a retail grocery shop to manage their services with ease of technology and minimize the errors which can be caused by paper work.

2.1. Business Situation:

Retail shops get numerous products (prod_id, name, price, sale_price, desc, mnf_date, exp_date, aisle_row_no) from various suppliers (supplier_id, name, phone, email, address). These products are displayed at different aisles (aisle_id, aisle_no, desc) and stock of all products being tracked in the inventories (inventory_id, prod_id, qty, in_stock, supplier_id) . A customer can pick up the products from these aisles and simply walk to the billing counter where Employee (emp_id, name, email, phone, position, Job_role, joining_date, leaving_date) of a retail shop can assist a customer (cust_id, name, email, phone, dob, address, postcode, age) to process the order (order_id, cust_id, emp_id, total_qty, price, tax, total_price, order_date, order_time). After billing all the order products (order_prod_id, price, qty, prod_id, order_id) customer can process a payment (payment_id, order_id, price, desc, status, method, updated_at)

2.2. Business Rules:

- Aisle may display many products.
- A customer can add multiple products in a single order.
- A single employee can assist various customers for the order billing process.
- There will be one payment instance for the order
- The stock quantity of all products will be tracked at the inventory i.e. Inventory will have collection of numerous products
- A Supplier will supply various products to a retail shop
- A customer can have multiple orders

2.3. List of Entity/Attributes:

1. Entity: CUSTOMERS:
   a. Attributes: cust_id, name, email, phone, dob, address, postcode, age
2. Entity: PRODUCTS:
   a. Attributes: prod_id, inventory_id, name, price, sale_price, desc, mnf_date, exp_date, aisle_row_no
3. Entity: EMPLOYEES:
   a. Attributes: emp_id, name, email, phone, position, job_role, joining_date, leaving_date
4. Entity: ORDERS:
   a. Attributes: order_id, cust_id, emp_id, total_qty, price, tax, total_price, order_date, order_time
5. Entity: ORDER_PRODUCTS:
   a. Attributes: order_prod_id, price, qty, prod_id, order_id
6. Entity: PAYMENTS:
   a. Attributes: payment_id, order_id, price, desc, status, method, updated_at
7. Entity: SUPPLIER:
   a. Attributes: supplier_id, name, phone, email, address
8. Entity: INVENTORIES:
   a. Attributes: inventory_id, qty, in_stock, supplier_id
9. Entity: AISLE_NO:
   a. Attributes: aisle_id, aisle_no, desc

2.4. Simple Relationships:

[CUSTOMERS] **1** ← HAS → **N** [ORDERS]

[PRODUCTS] **1** ← LISTS → **1** [INVENTORIES]

[EMPLOYEES] **1** ← ASSIST → **N** [ORDERS]

[ORDERS] **1** ← GENERATES → **1** [PAYMENTS]

[ORDERS] **1** ← HAS → **N** [ORDERS_PRODUCTS]

[SUPPLIERS] **M** ← SUPPLY → **N** [INVENTORIES]

[AISLE] **M** ← DISPLAYS → **N** [PRODUCTS]

2.5. Connectivity, Cardinalities and Participation:

A CUSTOMER can have minimum of __1__ ORDERS

A CUSTOMER can have maximum of __N__ ORDERS

Reverse:

ORDERS connect with minimum of __1__ CUSTOMERS

ORDERS connect with maximum of __1__ CUSTOMERS

A SUPPLIER supply products minimum of __1__ INVENTORY

A SUPPLIER supply products maximum of __N__ INVENTORIES

Reverse:

INVENTORY will be filled by minimum of __1__ Supplier

INVENTORY will be filled by maximum of __N__ SUPPLIERIES

.

An EMPLOYEE managing minimum of __1__ ORDER

An EMPLOYEE managing maximum of __N__ ORDERS

Reverse:

An ORDER is managed by minimum of __1__ EMPLOYEE

An ORDER is managed by maximum of __1__ EMPLOYEE

An ORDER generating minimum of __1__ PAYMENT

An ORDER generating maximum of __1__ PAYMENT

Reverse:

A PAYMENT is generated by minimum of __1__ ORDER

A PAYMENT is generated by maximum of __1__ ORDER

A PRODUCT is listed at minimum of _1_ INVENTORY

A PRODUCT is listed at Maximum of _1_ INVENTORY

Reverse:

An INVENTORY list minimum of _1_ PRODUCT

An INVENTORY list Maximum of _N_PRODUCT

An ORDER has minimum of _1_ ORDER_PRODUCTS

An ORDER has Maximum of _M_ ORDER_PRODUCTS

Reverse:

An ORDER_PRODUCTS has minimum of _1_ ORDER

An ORDER_PRODUCTS has Maximum of _M_ ORDER

An AISLE displays minimum of _1_ PRODUCTS

An AISLE displays Maximum of _N_ PRODUCTS

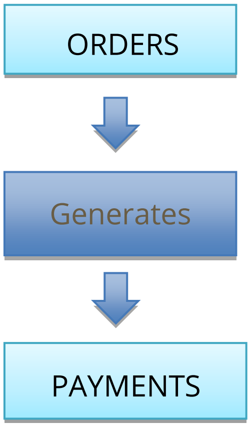
Reverse:

A PRODUCTS displayed at minimum of _1_ AISLE

A PRODUCTS displayed at Maximum of _N_ AISLE

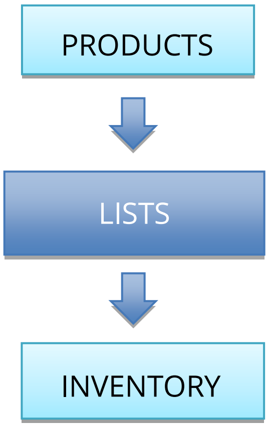
2.6 ERD Mapping:

2.6.1. Mapping 1: 1 Relationships:



order_id	cust_id	emp_id	qty	price	Tax	total_price	order_date	order_time
001	678	234	3	10.00	0.60	10.60	27-10-21	10:10
002	679	345	4	25.00	0.80	25.80	28-10-21	12:00
003	701	321	1	05.00	0.10	05.10	29-10-21	16:00

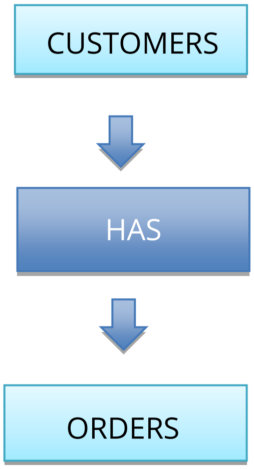
payment_id	order_id	price	status	method	updated_at
987	003	05.10	completed	COC	29-10-21 16:01
876	002	25.80	completed	Card	28-21 12:01
678	001	10.00	pending	Card	27-10-21 10:11



prod_id	name	price	sale_price	desc	mnf_date	exp_date	aisle_row_no
765	Bread	10	6.7	all types of breads	27/12/2021	30/12/2021	5
234	Cheese	11.5	10	shredded cheese	21/12/2021	30/12/2021	3
546	Drinks	12.4	11	cold drinks	15/12/2021	20/12/2021	6

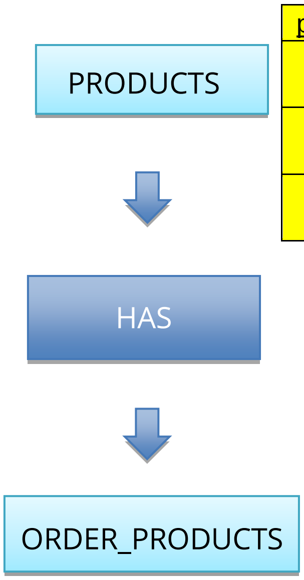
inventory_id	prod_id	qty	in_stock	supplier_id
788	765	230	TRUE	12
987	234	120	TRUE	13
546	546	421	TRUE	14

2.6.2. Mapping 1: N Relationships:



cust_id	name	email	phone	dob	address	postcode	age
001	Abi	abi@xyz.com	0123456789	10-04-1990	xyz	0123	15
002	Ola	ola@xyz.com	0287654567	18-06-1992	abh	1234	16
003	Eve	eve@xyz.com	0976546765	03-03-1993	pol	1245	17
004	Wed	wed@xyz.com	0456734568	19-09-1994	yus	6543	18

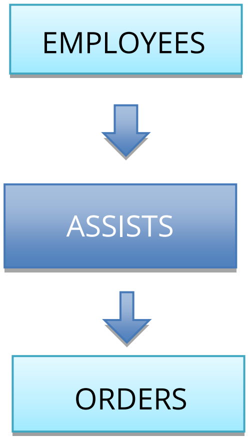
order_id	cust_id	emp_id	total_qty	price	tax	total_price	order_date	order_time
001	678	234	3	10	0.6	10.6	27/10/2021	10:10
002	679	345	4	25	0.8	25.8	28/10/2021	12:00
003	701	321	1	5	0.1	5.1	29/10/2021	16:00



prod_id	name	price	sale_price	desc	mnf_date	exp_date	aisle_row_no
765	Bread	10	6.7	all types of breads	27/12/2021	30/12/2021	5
234	Cheese	11.5	10	shredded cheese	21/12/2021	30/12/2021	3
546	Drinks	12.4	11	cold drinks	15/12/2021	20/12/2021	6

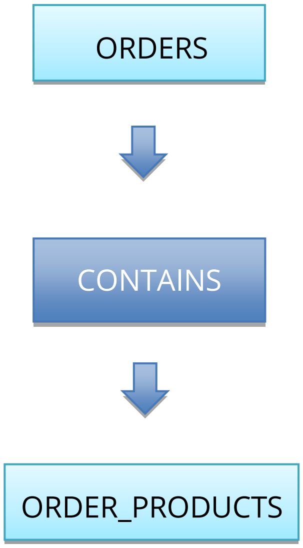
order_prod_id	price	qty	prod_id	order_id
456	10	3	765	001
879	25	4	234	002
986	5	1	546	003

emp_id	name	email	phone	position	Job_role	joining_date	leaving_date
234	Yem	yem@xyz.com	0987654567	staff	Retail_Assistant	27/10/2020	27/11/2020
345	Pop	pop@xyz.com	0875567787	staff	Cashier	28/10/2020	28/11/2020
321	Lad	lad@xyz.com	0345678766	manager	Staff_Management	29/10/2020	29/11/2020



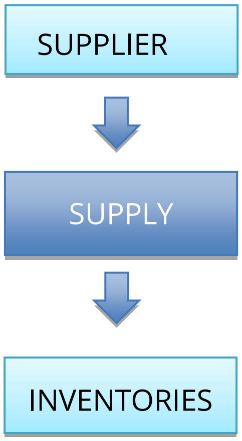
order_id	cust_id	emp_id	total_qty	price	tax	total_price	order_date	order_time
001	678	234	3	10	0.6	10.6	27/10/2021	10:10
002	679	345	4	25	0.8	25.8	28/10/2021	12:00
003	701	321	1	5	0.1	5.1	29/10/2021	16:00

order_id	cust_id	emp_id	total_qty	price	tax	total_price	order_date	order_time
001	678	234	3	10	0.6	10.6	27/10/2021	10:10
002	679	345	4	25	0.8	25.8	28/10/2021	12:00
003	701	321	1	5	0.1	5.1	29/10/2021	16:00



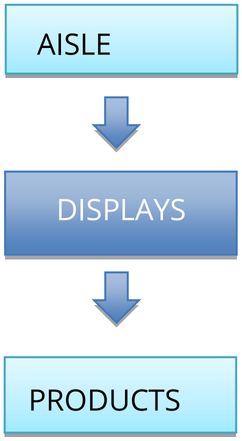
order_prod_id	price	qty	prod_id	order_id
456	10	3	765	001
879	25	4	234	002
986	5	1	546	003

2.6.3. Mapping M: N Relationships:



supplier_id	name	phone	Email	address
12	SUPP1	0987687678	supp1@xyz.com	sup add
13	SUPP2	0987658778	supp2@xyz.com	supp2 add
14	SUPP3	0453434534	supp3@xyz.com	supp3 add

inventory_id	prod_id	qty	in_stock	supplier_id
788	765	230	TRUE	12
987	234	120	TRUE	13
546	546	421	TRUE	14



aisle_id	aisle_no	Desc
098	5	Bread Section
076	3	Cheese Section
061	6	Drink Section

prod_id	name	price	sale_price	desc	mnf_date	exp_date	aisle_row_no
765	Bread	10	6.7	all types of breads	27/12/2021	30/12/2021	5
234	Cheese	11.5	10	shredded cheese	21/12/2021	30/12/2021	3
546	Drinks	12.4	11	cold drinks	15/12/2021	20/12/2021	6

3. Database Design:

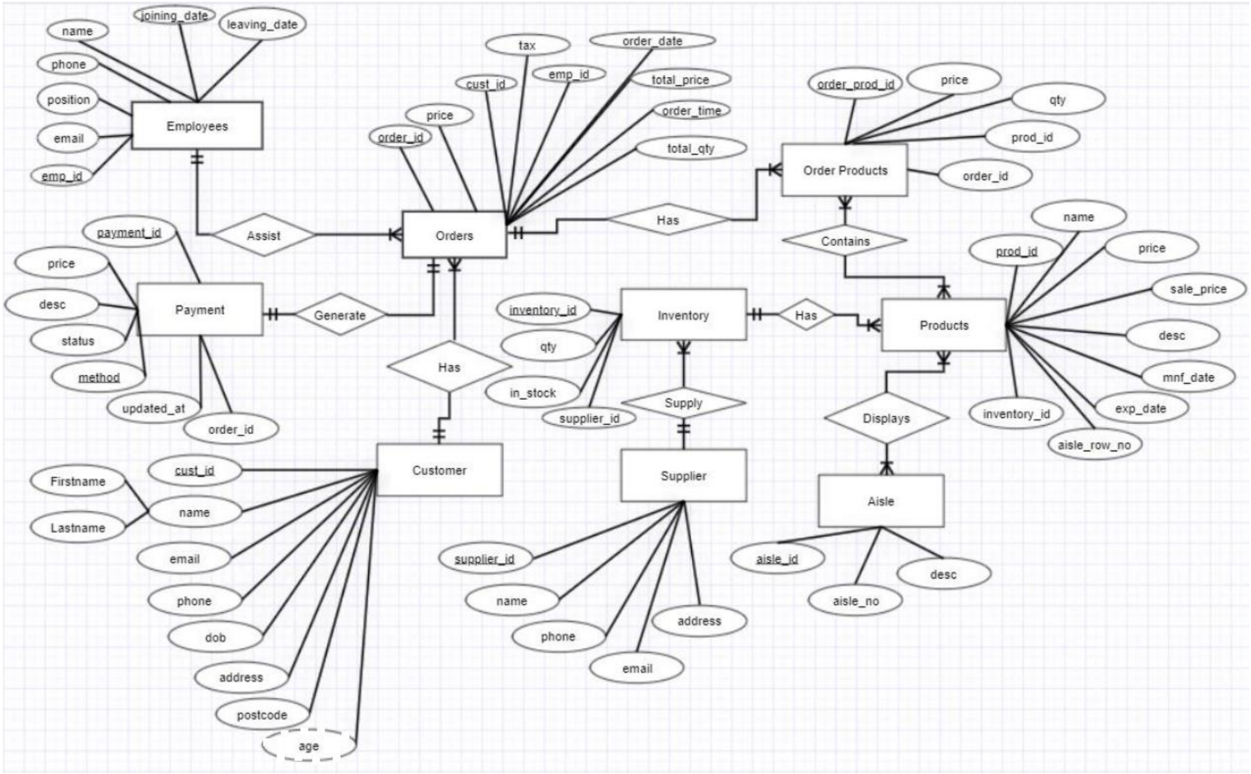


Fig 1: ER Diagram

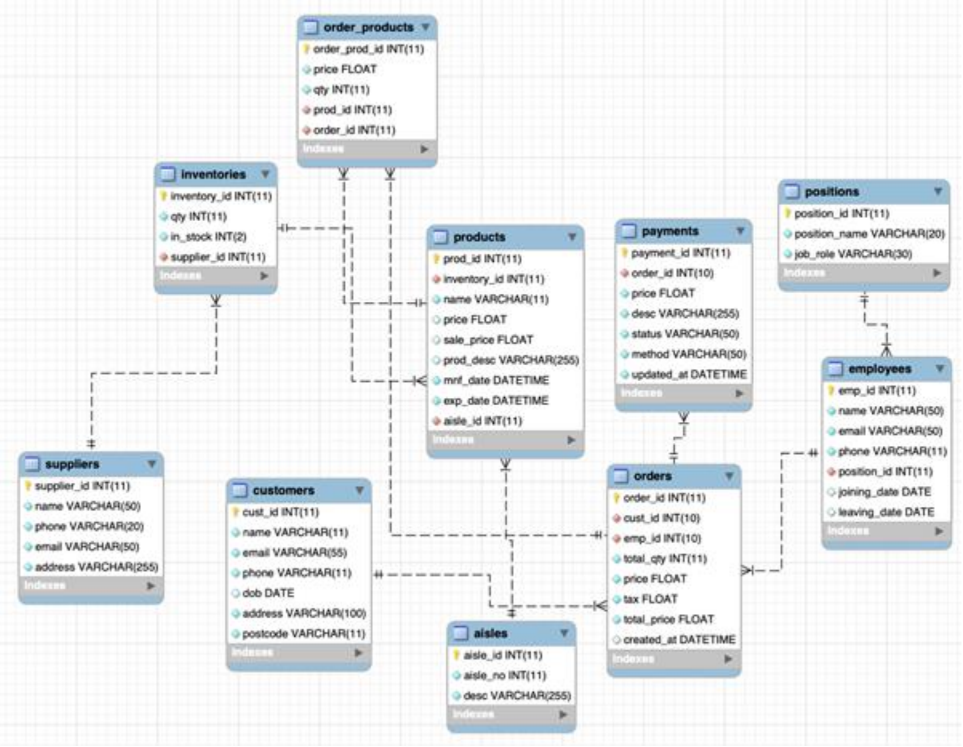


Fig 2: Class Diagram after normalization

4. Normalization

Whenever any table really not properly normalized as well as has duplicate information, this would not only consume more disk space, but this will be tough to organize as well as update into the DB without experiencing data loss.

4.1 Customers

cust_id	name	email	phone	dob	address	postcode	age
001	Abi	abi@xyz.com	0123456789	10-04-1990	xyz	0123	15
002	Ola	ola@xyz.com	0287654567	18-06-1992	abh	1234	16
003	Eve	eve@xyz.com	0976546765	03-03-1993	pol	1245	17

4.2 Products

prod_id	Name	price	sale_price	desc	mnf_date	exp_date	aisle_row_no
765	Bread	10	6.7	all types of breads	27/12/2021	30/12/2021	5
234	Cheese	11.5	10	shredded cheese	21/12/2021	30/12/2021	3
546	Drinks	12.4	11	cold drinks	15/12/2021	20/12/2021	6

4.3 Inventories:

inventory_id	prod_id	qty	in_stock	supplier_id
788	765	230	TRUE	12
987	234	120	TRUE	13
546	546	421	TRUE	14

4.4 Order_Products

order_prod_id	price	qty	prod_id	order_id
456	10	3	765	001
879	25	4	234	002
986	5	1	546	003

4.5 Suppliers:

supplier_id	name	phone	Email	address
12	SUPP1	0987687678	supp1@xyz.com	sup add
13	SUPP2	0987658778	supp2@xyz.com	supp2 add
14	SUPP3	0453434534	supp3@xyz.com	supp3 add

4.6 Payments:

payment_id	order_id	price	status	method	updated_at
987	003	05.10	completed	COC	29-10-21 16:01
876	002	25.80	completed	Card	28-21 12:01
678	001	10.00	pending	Card	27-10-21 10:11

4.7 Aisle_No

aisle_id	aisle_no	Desc
098	5	Bread Section
076	3	Cheese Section
061	6	Drink Section

4.8 Orders:

order_id	cust_id	emp_id	total_qty	price	tax	total_price	order_date	order_time
001	678	234	3	10	0.6	10.6	27/10/2021	10:10
002	679	345	4	25	0.8	25.8	28/10/2021	12:00
003	701	321	1	5	0.1	5.1	29/10/2021	16:00

The above tables have been in the 3NF because each column has just the same type of one value, every column has a distinct name, as well as the sequence in which the data is stored is not important. In addition, the main key (PK) has been determined, and all non-key properties are totally entirely reliant mostly on PK. Therefore, there seem to be no transitive or partial dependencies.

4.9 Employees

emp_id	name	email	phone	position	Job_role	joining_date	leaving_date
234	Yem	yem@xyz.com	0987654567	staff	Retail_Assistant	27/10/2020	27/11/2020
345	Pop	pop@xyz.com	0875567787	staff	Cashier	28/10/2020	28/11/2020
321	Lad	lad@xyz.com	0345678766	manager	Staff_Management	29/10/2020	29/11/2020

The employee table has a partial dependency, to solve this position column and job role has to take out in the new separate table.

emp_id	name	email	phone	Position_id	joining_date	leaving_date
234	Yem	yem@xyz.com	0987654567	1	27/10/2020	27/11/2020
345	Pop	pop@xyz.com	0875567787	2	28/10/2020	28/11/2020
321	Lad	lad@xyz.com	0345678766	3	29/10/2020	29/11/2020

Position_id	Position_Name	Job_role
1	staff	Retail_Assistant
2	staff	Cashier
3	manager	Staff_Management

5. Database Implementation:

5.1. Table Creation:

Student (Aditya: 21144695)

Customers Table:

```
CREATE TABLE `ad_project`.`customers` ( `cust_id` INT
UNSIGNED NOT NULL AUTO_INCREMENT , `name` VARCHAR(11) NOT NULL , `email` VARCHAR(11) N
OT NULL , `phone` VARCHAR(11) NOT NULL , `dob` DATE NULL DEFAULT NULL , `address` VARC
HAR(11) NOT NULL , `postcode` VARCHAR(11) NOT NULL
, PRIMARY KEY (`cust_id`)) ENGINE = InnoDB;

ALTER TABLE `customers` ADD UNIQUE(`email`);
```

Student (Aditya: 21144695)

Products Table:

```
CREATE TABLE `ad_project`.`products` ( `prod_id` INT
UNSIGNED NOT NULL AUTO_INCREMENT , `inventory_id` INT NOT NULL , `name` VARCHAR(11) NO
T NULL , `price` FLOAT(24) NULL , `sale_price` FLOAT(24) NULL , `prod_desc` VARCHAR(25
5) NULL DEFAULT NULL , `mnf_date` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP , `exp_d
ate` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP , `aisle_id` INT(11) UNSIGNED NOT NUL
L , PRIMARY KEY (`prod_id`)) ENGINE = InnoDB;

ALTER TABLE `products` ADD CONSTRAINT `aisles_has_products` FOREIGN KEY (`aisle_id`)
REFERENCES `aisles`(`aisle_id`) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE `products` CHANGE `inventory_id` `inventory_id` INT(11) UNSIGNED NOT NULL;

ALTER TABLE `products` ADD CONSTRAINT `inventory_products` FOREIGN KEY
(`inventory_id`) REFERENCES `inventories`(`inventory_id`) ON DELETE NO ACTION ON
UPDATE NO ACTION;
```

Student (Aditya: 21144695)

Employees Table:

```
CREATE TABLE `ad_project`.`employees` ( `emp_id` INT
UNSIGNED NOT NULL AUTO_INCREMENT , `name` VARCHAR(50) NOT NULL , `email` VARCHAR(50) N
OT NULL , `phone` VARCHAR(11) NOT NULL , `position_id` INT(25) NOT NULL , `joining_dat
e` DATE NULL , `leaving_date` DATE NULL , PRIMARY KEY (`emp_id`)) ENGINE = InnoDB;

ALTER TABLE `employees` ADD CONSTRAINT `employee_position_id` FOREIGN KEY
(`position_id`) REFERENCES `positions`(`position_id`) ON DELETE NO ACTION ON UPDATE NO
ACTION;
```

Student (Aditya: 21144695)

Positions Table:

```
CREATE TABLE `ad_project`.`positions` ( `Position_id` int(11) NOT NULL
AUTO_INCREMENT ,`Position_Name` varchar(20) NOT NULL,`Job_role` varchar(30) NOT NULL
, PRIMARY KEY (`Position_id`)) ENGINE=InnoDB;
```

Student (Aditya: 21144695)

Orders Table:

```
CREATE TABLE `ad_project`.`orders` ( `order_id` INT
UNSIGNED NOT NULL AUTO_INCREMENT , `cust_id` INT UNSIGNED NOT NULL , `emp_id` INT UNSI
GNED NOT NULL , `total_qty` INT NOT NULL , `price` FLOAT NOT NULL , `tax` FLOAT NOT NU
LL , `total_price` FLOAT NOT NULL , `order_datetime` DATETIME NOT
NULL DEFAULT CURRENT_TIMESTAMP, PRIMARY KEY (`order_id`)) ENGINE = InnoDB;
```

```
ALTER TABLE `orders` ADD CONSTRAINT `order_customers` FOREIGN KEY (`cust_id`)
 REFERENCES `customers`(`cust_id`) ON DELETE NO ACTION ON UPDATE NO ACTION; A
LTER TABLE `orders` ADD CONSTRAINT `order_employees` FOREIGN KEY (`emp_id`) R
EFERENCES `employees`(`emp_id`) ON DELETE NO ACTION ON UPDATE NO ACTION;
```

Student (Darshan: 21171488)

Order Products Table:

```
CREATE TABLE `ad_project`.`order_products` ( `order_prod_id` INT NOT NULL AUTO_INCREME
NT , `price` FLOAT NOT NULL , `qty` INT NOT NULL , `prod_id` INT UNSIGNED NOT NULL , `
order_id` INT UNSIGNED NOT NULL , PRIMARY KEY (`order_prod_id`)) ENGINE = InnoDB;
```

```
ALTER TABLE `order_products` ADD CONSTRAINT `op_product` FOREIGN KEY (`prod_id`)
REFERENCES `products`(`prod_id`) ON DELETE NO ACTION ON UPDATE NO ACTION; ALTER TABLE
`order_products` ADD CONSTRAINT `op_order` FOREIGN KEY (`order_id`) REFERENCES
`orders`(`order_id`) ON DELETE NO ACTION ON UPDATE NO ACTION;
```

Student (Darshan: 21171488)

Payments Table:

```
CREATE TABLE `ad_project`.`payments` ( `payment_id` INT NOT NULL AUTO_INCREMENT , `ord
er_id` INT UNSIGNED NOT NULL , `price` FLOAT NOT NULL , `desc` VARCHAR(255) NOT NULL ,
 `status` VARCHAR(50) NOT NULL , `method` VARCHAR(50) NOT NULL , `updated_at` DATETIME
 NOT NULL DEFAULT CURRENT_TIMESTAMP , PRIMARY KEY (`payment_id`)) ENGINE = InnoDB;
```

```
ALTER TABLE `payments` ADD CONSTRAINT `order_payments` FOREIGN KEY (`order_id
`) REFERENCES `orders`(`order_id`) ON DELETE NO ACTION ON UPDATE NO ACTION;
```

Student (Darshan: 21171488)

Suppliers Table:

```
CREATE TABLE `ad_project`.`suppliers` ( `supplier_id`
UNSIGNED INT NOT NULL AUTO_INCREMENT , `name` VARCHAR(50) NOT NULL , `phone` INT(11)
NOT NULL , `email` VARCHAR(50) NOT NULL , `address` VARCHAR(255) NOT NULL , PRIMARY KE
Y (`supplier_id`)) ENGINE = InnoDB;
```

```
ALTER TABLE ` suppliers ` ADD CONSTRAINT `supplier_inventories` FOREIGN KEY
(`supplier_id`) REFERENCES `inventories`(`supplier_id`) ON DELETE NO ACTION ON UPDATE
NO ACTION;
```

Student (Darshan: 21171488)

Inventories Table:

```
CREATE TABLE `ad_project`.`inventories` ( `inventory_id` INT NOT NULL AUTO_INCREMENT ,
 `qty` INT NOT NULL , `in_stock` INT(2) NOT NULL COMMENT '0 for out of stock, 1 for in
stock' , `supplier_id` INT UNSIGNED NOT NULL , PRIMARY KEY (`inventory_id`)) ENGI
```

```
ALTER TABLE `inventories` CHANGE `inventory_id` `inventory_id` INT(11) UNSIGNED NOT NU
LL AUTO_INCREMENT;
```

```
ALTER TABLE `inventories` ADD CONSTRAINT `supplier_inventories` FOREIGN KEY (
`supplier_id`) REFERENCES `suppliers`(`supplier_id`) ON DELETE NO ACTION ON U
PDATE NO ACTION;
```

Student (Darshan: 21171488)

Aisles Table:

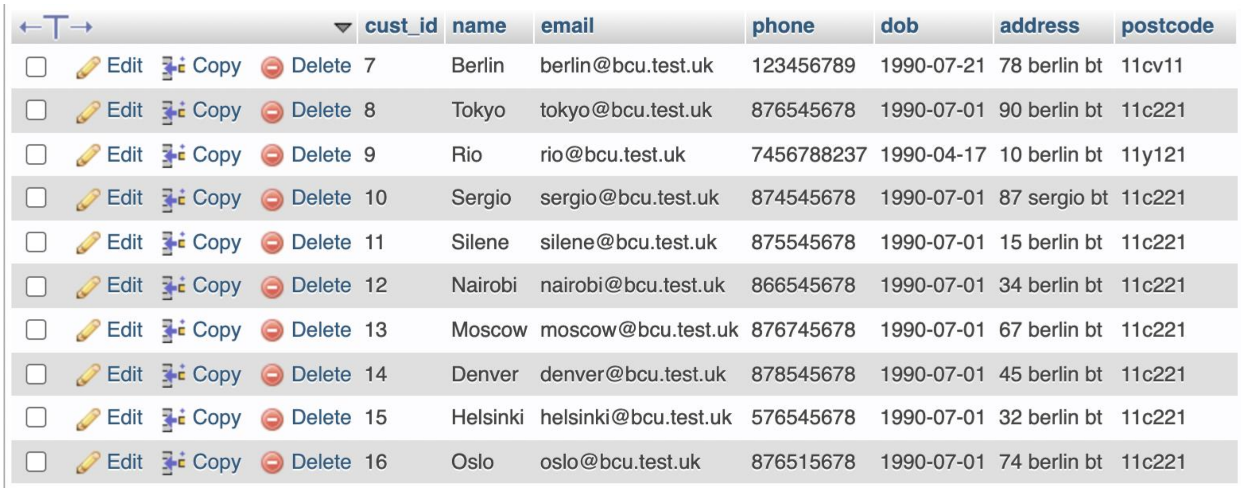
```
CREATE TABLE `ad_project`.`aisles` ( `aisle_id` INT NOT NULL AUTO_INCREMENT , `aisle_n
o` INT(11) UNSIGNED
NOT NULL , `desc` VARCHAR(255) NOT NULL , PRIMARY KEY (`aisle_id`)) ENGINE = InnoDB;
```

```
ALTER TABLE `aisles` CHANGE `aisle_id` `aisle_id` INT(11) UNSIGNED NOT NULL AUTO_INCRE
MENT;
```

5.2. Data entry in tables:

Customers:

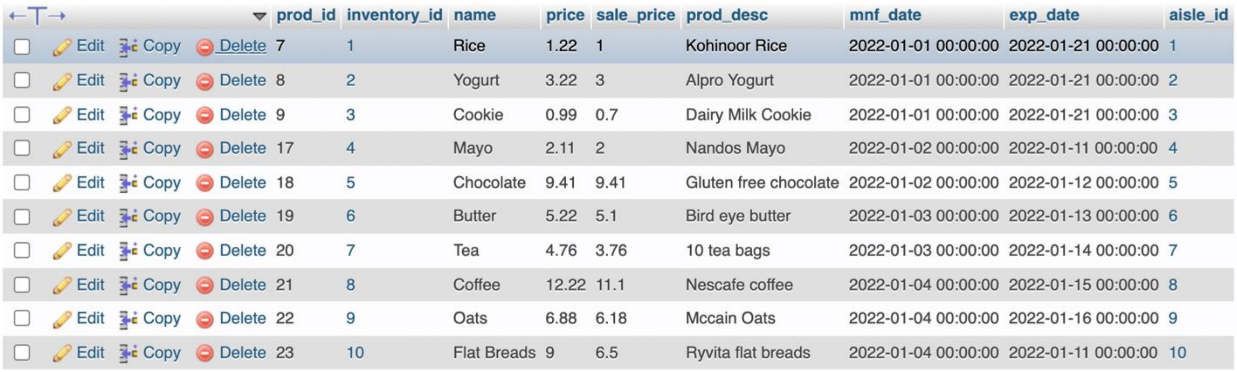
```
INSERT INTO customers
    (`cust_id`,`name`,`email`,`phone`,`dob`,`address`,`postcode`)
VALUES
    (NULL, 'Berlin', 'berlin@bcu.test.uk',0123456789,'1990-07-21','78 berlin bt',
'11cv11'),
    (NULL, 'Tokyo', 'tokyo@bcu.test.uk',0876545678,'1990-07-01','90 berlin bt',
'11c221'),
    (NULL, 'Sergio', 'sergio@bcu.test.uk',0874545678,'1990-07-01','87 sergio bt',
'11c221'),
    (NULL, 'Silene', 'silene@bcu.test.uk',0875545678,'1990-07-01','15 berlin bt',
'11c221'),
    (NULL, 'Nairobi', 'nairobi@bcu.test.uk',0866545678,'1990-07-01','34 berlin bt',
'11c221'),
    (NULL, 'Moscow', 'moscow@bcu.test.uk',0876745678,'1990-07-01','67 berlin bt',
'11c221'),
    (NULL, 'Denver', 'denver@bcu.test.uk',0878545678,'1990-07-01','45 berlin bt',
'11c221'),
    (NULL, 'Helsinki', 'helsinki@bcu.test.uk',0576545678,'1990-07-01','32 berlin bt',
'11c221'),
    (NULL, 'Oslo', 'oslo@bcu.test.uk',0876515678,'1990-07-01','74 berlin bt',
'11c221'),
    (NULL, 'Rio', 'rio@bcu.test.uk',7456788237,'1990-04-17','15 berlin bt', '11y121');
```



←T→			▼	cust_id	name	email	phone	dob	address	postcode
□	Edit	Copy	Delete	7	Berlin	berlin@bcu.test.uk	123456789	1990-07-21	78 berlin bt	11cv11
□	Edit	Copy	Delete	8	Tokyo	tokyo@bcu.test.uk	876545678	1990-07-01	90 berlin bt	11c221
□	Edit	Copy	Delete	9	Rio	rio@bcu.test.uk	7456788237	1990-04-17	10 berlin bt	11y121
□	Edit	Copy	Delete	10	Sergio	sergio@bcu.test.uk	874545678	1990-07-01	87 sergio bt	11c221
□	Edit	Copy	Delete	11	Silene	silene@bcu.test.uk	875545678	1990-07-01	15 berlin bt	11c221
□	Edit	Copy	Delete	12	Nairobi	nairobi@bcu.test.uk	866545678	1990-07-01	34 berlin bt	11c221
□	Edit	Copy	Delete	13	Moscow	moscow@bcu.test.uk	876745678	1990-07-01	67 berlin bt	11c221
□	Edit	Copy	Delete	14	Denver	denver@bcu.test.uk	878545678	1990-07-01	45 berlin bt	11c221
□	Edit	Copy	Delete	15	Helsinki	helsinki@bcu.test.uk	576545678	1990-07-01	32 berlin bt	11c221
□	Edit	Copy	Delete	16	Oslo	oslo@bcu.test.uk	876515678	1990-07-01	74 berlin bt	11c221

Products:

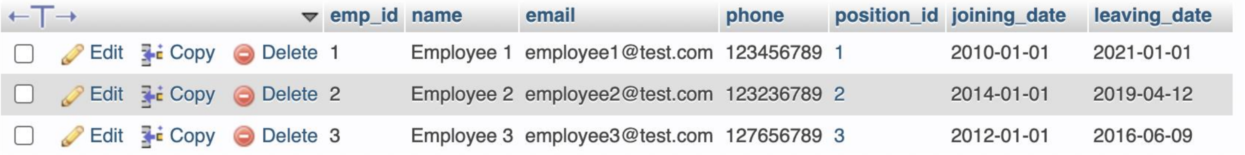
```
INSERT INTO products

(`prod_id`,`inventory_id`,`name`,`price`,`sale_price`,`prod_desc`,`mnf_date`,`exp_date
`,`aisle_id`)
VALUES
    (NULL, 1, 'Rice',1.22,1.00,'Kohinoor Rice','2022-01-01','2022-01-21', 1),
    (NULL, 2, 'Yogurt',3.22,3.00,'Alpro Yogurt','2022-01-01','2022-01-21', 2),
    (NULL, 3, 'Cookie',0.99,0.70,'Dairy Milk Cookie','2022-01-01','2022-01-21', 3),
    (NULL, 4, 'Mayo',2.11,2.00,'Nandos Mayo','2022-01-02','2022-01-11', 3),
    (NULL, 5, 'Chocolate',9.41,9.41,'Gluten free chocolate','2022-01-02','2022-01-12',
3),
    (NULL, 6, 'Butter',5.22,5.10,'Bird eye butter','2022-01-03','2022-01-13', 3),
    (NULL, 7, 'Tea',4.76,3.76,'10 tea bags','2022-01-03','2022-01-14', 3),
    (NULL, 8, 'Coffee',12.22,11.10,'Nescafe coffee','2022-01-04','2022-01-15', 3),
    (NULL, 9, 'Oats',6.88,6.18,'Mccain Oats','2022-01-04','2022-01-16', 3),
    (NULL, 10, 'Flat Breads',9.00,6.50,'Ryvita flat breads','2022-01-04','2022-01-11',
3);
```



←T→			▼	prod_id	inventory_id	name	price	sale_price	prod_desc	mnf_date	exp_date	aisle_id
□	Edit	Copy	Delete	7	1	Rice	1.22	1	Kohinoor Rice	2022-01-01 00:00:00	2022-01-21 00:00:00	1
□	Edit	Copy	Delete	8	2	Yogurt	3.22	3	Alpro Yogurt	2022-01-01 00:00:00	2022-01-21 00:00:00	2
□	Edit	Copy	Delete	9	3	Cookie	0.99	0.7	Dairy Milk Cookie	2022-01-01 00:00:00	2022-01-21 00:00:00	3
□	Edit	Copy	Delete	17	4	Mayo	2.11	2	Nandos Mayo	2022-01-02 00:00:00	2022-01-11 00:00:00	4
□	Edit	Copy	Delete	18	5	Chocolate	9.41	9.41	Gluten free chocolate	2022-01-02 00:00:00	2022-01-12 00:00:00	5
□	Edit	Copy	Delete	19	6	Butter	5.22	5.1	Bird eye butter	2022-01-03 00:00:00	2022-01-13 00:00:00	6
□	Edit	Copy	Delete	20	7	Tea	4.76	3.76	10 tea bags	2022-01-03 00:00:00	2022-01-14 00:00:00	7
□	Edit	Copy	Delete	21	8	Coffee	12.22	11.1	Nescafe coffee	2022-01-04 00:00:00	2022-01-15 00:00:00	8
□	Edit	Copy	Delete	22	9	Oats	6.88	6.18	Mccain Oats	2022-01-04 00:00:00	2022-01-16 00:00:00	9
□	Edit	Copy	Delete	23	10	Flat Breads	9	6.5	Ryvita flat breads	2022-01-04 00:00:00	2022-01-11 00:00:00	10

Employees:

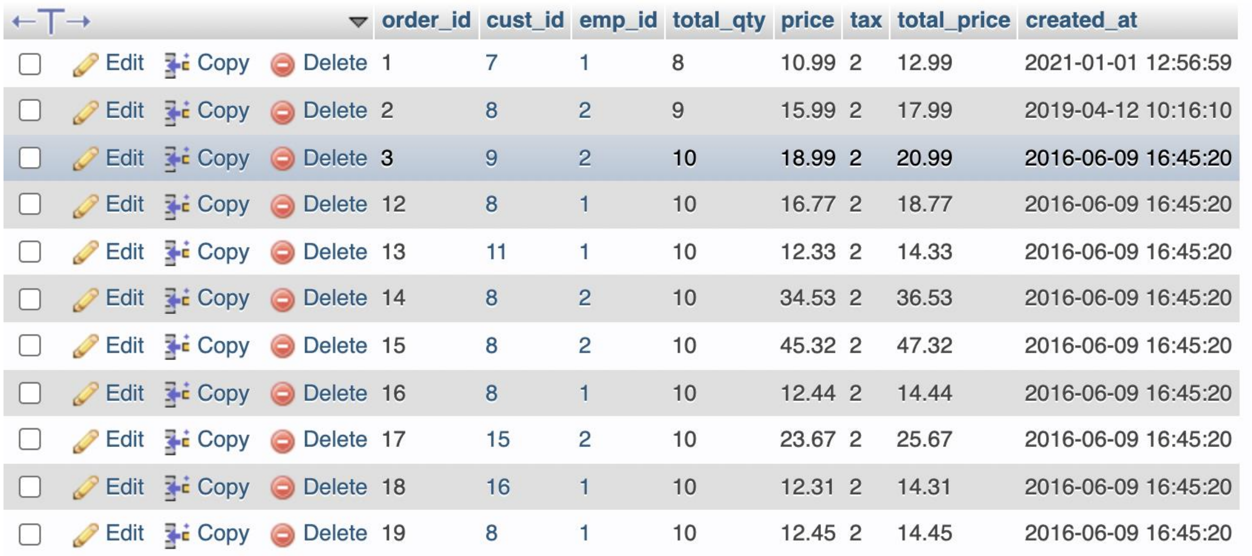
```
INSERT INTO employees
(`emp_id`,`name`,`email`,`phone`,`position_id`,`joining_date`,`leaving_date`)
VALUES
(NULL, 'Employee 1', 'employee1@test.com', 0123456789, 1, '2010-01-01', '2021-01-01'),
(NULL, 'Employee 2', 'employee2@test.com', 0123236789, 2, '2014-01-01', '2019-04-12'),
(NULL, 'Employee 3', 'employee3@test.com', 0127656789, 3, '2012-01-01', '2016-06-09');
```



←T→			▼	emp_id	name	email	phone	position_id	joining_date	leaving_date
□	Edit	Copy	Delete	1	Employee 1	employee1@test.com	123456789	1	2010-01-01	2021-01-01
□	Edit	Copy	Delete	2	Employee 2	employee2@test.com	123236789	2	2014-01-01	2019-04-12
□	Edit	Copy	Delete	3	Employee 3	employee3@test.com	127656789	3	2012-01-01	2016-06-09

Orders:

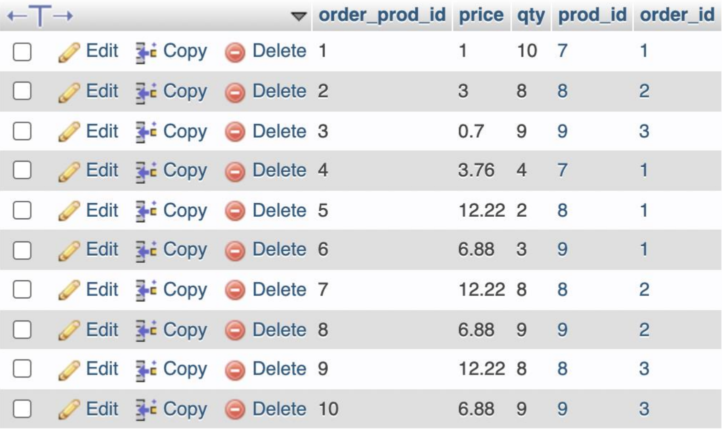
```
INSERT INTO orders
(`order_id`,`cust_id`,`emp_id`,`total_qty`,`price`,`tax`,`total_price`,`created_at`)
VALUES
(NULL, 7, 1, 8, 10.99, 2.00, 12.99, '2021-01-01 12:56:59'),
(NULL, 8, 2, 9, 15.99, 2.00, 17.99, '2019-04-12 10:16:10'),
(NULL, 9, 2, 10, 18.99, 2.00, 20.99, '2016-06-09 16:45:20'),
(NULL, 8, 1, 10, 16.77, 2.00, 18.77, '2016-06-09 16:45:20'),
(NULL, 11, 1, 10, 12.33, 2.00, 14.33, '2016-06-09 16:45:20'),
(NULL, 8, 2, 10, 34.53, 2.00, 36.53, '2016-06-09 16:45:20'),
(NULL, 8, 2, 10, 45.32, 2.00, 47.32, '2016-06-09 16:45:20'),
(NULL, 8, 1, 10, 12.44, 2.00, 14.44, '2016-06-09 16:45:20'),
(NULL, 15, 2, 10, 23.67, 2.00, 25.67, '2016-06-09 16:45:20'),
(NULL, 16, 1, 10, 12.31, 2.00, 14.31, '2016-06-09 16:45:20'),
(NULL, 8, 1, 10, 12.45, 2.00, 14.45, '2016-06-09 16:45:20');
```



←T→			▼	order_id	cust_id	emp_id	total_qty	price	tax	total_price	created_at
☐	Edit	Copy	Delete	1	7	1	8	10.99	2	12.99	2021-01-01 12:56:59
☐	Edit	Copy	Delete	2	8	2	9	15.99	2	17.99	2019-04-12 10:16:10
☐	Edit	Copy	Delete	3	9	2	10	18.99	2	20.99	2016-06-09 16:45:20
☐	Edit	Copy	Delete	12	8	1	10	16.77	2	18.77	2016-06-09 16:45:20
☐	Edit	Copy	Delete	13	11	1	10	12.33	2	14.33	2016-06-09 16:45:20
☐	Edit	Copy	Delete	14	8	2	10	34.53	2	36.53	2016-06-09 16:45:20
☐	Edit	Copy	Delete	15	8	2	10	45.32	2	47.32	2016-06-09 16:45:20
☐	Edit	Copy	Delete	16	8	1	10	12.44	2	14.44	2016-06-09 16:45:20
☐	Edit	Copy	Delete	17	15	2	10	23.67	2	25.67	2016-06-09 16:45:20
☐	Edit	Copy	Delete	18	16	1	10	12.31	2	14.31	2016-06-09 16:45:20
☐	Edit	Copy	Delete	19	8	1	10	12.45	2	14.45	2016-06-09 16:45:20

Order Products:

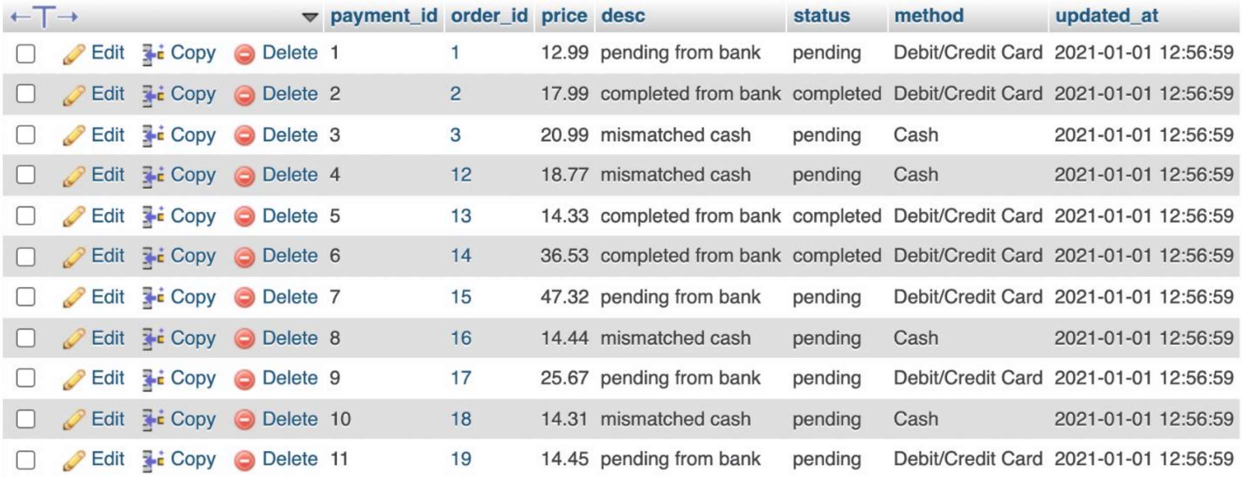
```
INSERT INTO order_products
(`order_prod_id`,`price`,`qty`,`prod_id`,`order_id`)
VALUES
(NULL, 1.00, 10, 7, 1),
(NULL, 3.00, 8, 8, 2),
(NULL, 0.70, 9, 9, 3),
(NULL, 3.76, 4, 7, 1),
(NULL, 12.22, 2, 8, 1),
(NULL, 6.88, 3, 9, 1),
(NULL, 12.22, 8, 8, 2),
(NULL, 6.88, 9, 9, 2),
(NULL, 12.22, 8, 8, 3),
(NULL, 6.88, 9, 9, 3);
```



←T→			▼	order_prod_id	price	qty	prod_id	order_id
□	Edit	Copy	Delete	1	1	10	7	1
□	Edit	Copy	Delete	2	3	8	8	2
□	Edit	Copy	Delete	3	0.7	9	9	3
□	Edit	Copy	Delete	4	3.76	4	7	1
□	Edit	Copy	Delete	5	12.22	2	8	1
□	Edit	Copy	Delete	6	6.88	3	9	1
□	Edit	Copy	Delete	7	12.22	8	8	2
□	Edit	Copy	Delete	8	6.88	9	9	2
□	Edit	Copy	Delete	9	12.22	8	8	3
□	Edit	Copy	Delete	10	6.88	9	9	3

Payment:

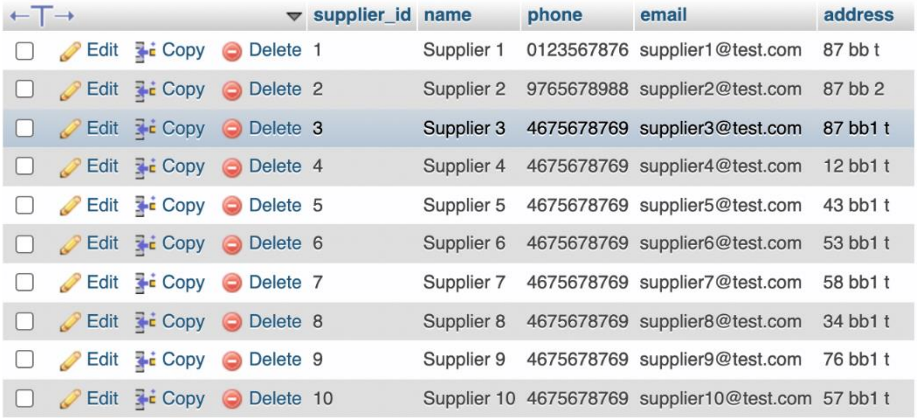
```
INSERT INTO payments
(`payment_id`,`order_id`,`price`,`desc`,`status`,`method`,`updated_at`)
VALUES
(NULL, 1, 12.99, 'pending from bank', 'pending', 'Debit/Credit Card', '2021-01-01
12:56:59'),
(NULL, 2, 17.99, 'completed from bank', 'completed', 'Debit/Credit Card', '2021-01-01
12:56:59'),
(NULL, 3, 20.99, 'mismatched cash', 'pending', 'Cash', '2021-01-01 12:56:59'),
(NULL, 12, 18.77, 'mismatched cash', 'pending', 'Cash', '2021-01-01 12:56:59'),
(NULL, 13, 14.33, 'completed from bank', 'completed', 'Debit/Credit Card', '2021-01-01
12:56:59'),
(NULL, 14, 36.53, 'completed from bank', 'completed', 'Debit/Credit Card', '2021-01-01
12:56:59'),
(NULL, 15, 47.32, 'pending from bank', 'pending', 'Debit/Credit Card', '2021-01-01
12:56:59'),
(NULL, 16, 14.44, 'mismatched cash', 'pending', 'Cash', '2021-01-01 12:56:59'),
(NULL, 17, 25.67, 'pending from bank', 'pending', 'Debit/Credit Card', '2021-01-01
12:56:59'),
(NULL, 18, 14.31, 'mismatched cash', 'pending', 'Cash', '2021-01-01 12:56:59'),
(NULL, 19, 14.45, 'pending from bank', 'pending', 'Debit/Credit Card', '2021-01-01
12:56:59');
```



←T→				payment_id	order_id	price	desc	status	method	updated_at
□	Edit	Copy	Delete	1	1	12.99	pending from bank	pending	Debit/Credit Card	2021-01-01 12:56:59
□	Edit	Copy	Delete	2	2	17.99	completed from bank	completed	Debit/Credit Card	2021-01-01 12:56:59
□	Edit	Copy	Delete	3	3	20.99	mismatched cash	pending	Cash	2021-01-01 12:56:59
□	Edit	Copy	Delete	4	12	18.77	mismatched cash	pending	Cash	2021-01-01 12:56:59
□	Edit	Copy	Delete	5	13	14.33	completed from bank	completed	Debit/Credit Card	2021-01-01 12:56:59
□	Edit	Copy	Delete	6	14	36.53	completed from bank	completed	Debit/Credit Card	2021-01-01 12:56:59
□	Edit	Copy	Delete	7	15	47.32	pending from bank	pending	Debit/Credit Card	2021-01-01 12:56:59
□	Edit	Copy	Delete	8	16	14.44	mismatched cash	pending	Cash	2021-01-01 12:56:59
□	Edit	Copy	Delete	9	17	25.67	pending from bank	pending	Debit/Credit Card	2021-01-01 12:56:59
□	Edit	Copy	Delete	10	18	14.31	mismatched cash	pending	Cash	2021-01-01 12:56:59
□	Edit	Copy	Delete	11	19	14.45	pending from bank	pending	Debit/Credit Card	2021-01-01 12:56:59

Supplier:

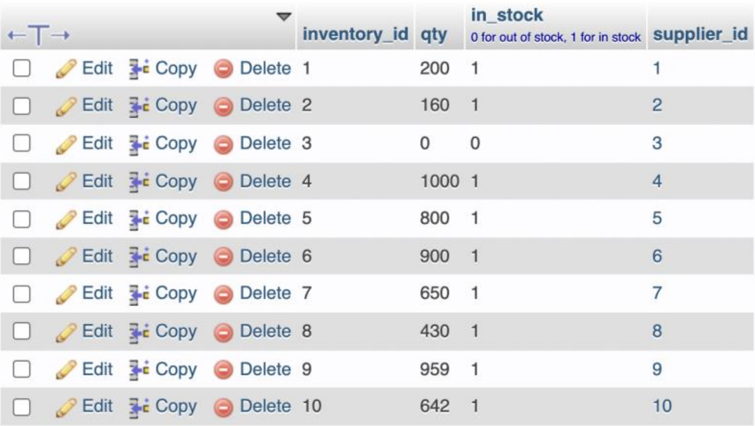
```
INSERT INTO suppliers
    (`supplier_id`,`name`,`phone`,`email`,`address`)
VALUES
    (1, 'Supplier 1','0123567876', 'supplier1@test.com', '87 bb t'),
    (2, 'Supplier 2','9765678988', 'supplier2@test.com', '87 bb 2'),
    (3, 'Supplier 3','4675678769', 'supplier3@test.com', '87 bb1 t'),
    (4, 'Supplier 4','4675678769', 'supplier4@test.com', '12 bb1 t'),
    (5, 'Supplier 5','4675678769', 'supplier5@test.com', '43 bb1 t'),
    (6, 'Supplier 6','4675678769', 'supplier6@test.com', '53 bb1 t'),
    (7, 'Supplier 7','4675678769', 'supplier7@test.com', '58 bb1 t'),
    (8, 'Supplier 8','4675678769', 'supplier8@test.com', '34 bb1 t'),
    (9, 'Supplier 9','4675678769', 'supplier9@test.com', '76 bb1 t'),
    (10, 'Supplier 10','4675678769', 'supplier10@test.com', '57 bb1 t');
```



←T→				supplier_id	name	phone	email	address
□	Edit	Copy	Delete	1	Supplier 1	0123567876	supplier1@test.com	87 bb t
□	Edit	Copy	Delete	2	Supplier 2	9765678988	supplier2@test.com	87 bb 2
□	Edit	Copy	Delete	3	Supplier 3	4675678769	supplier3@test.com	87 bb1 t
□	Edit	Copy	Delete	4	Supplier 4	4675678769	supplier4@test.com	12 bb1 t
□	Edit	Copy	Delete	5	Supplier 5	4675678769	supplier5@test.com	43 bb1 t
□	Edit	Copy	Delete	6	Supplier 6	4675678769	supplier6@test.com	53 bb1 t
□	Edit	Copy	Delete	7	Supplier 7	4675678769	supplier7@test.com	58 bb1 t
□	Edit	Copy	Delete	8	Supplier 8	4675678769	supplier8@test.com	34 bb1 t
□	Edit	Copy	Delete	9	Supplier 9	4675678769	supplier9@test.com	76 bb1 t
□	Edit	Copy	Delete	10	Supplier 10	4675678769	supplier10@test.com	57 bb1 t

Inventory:

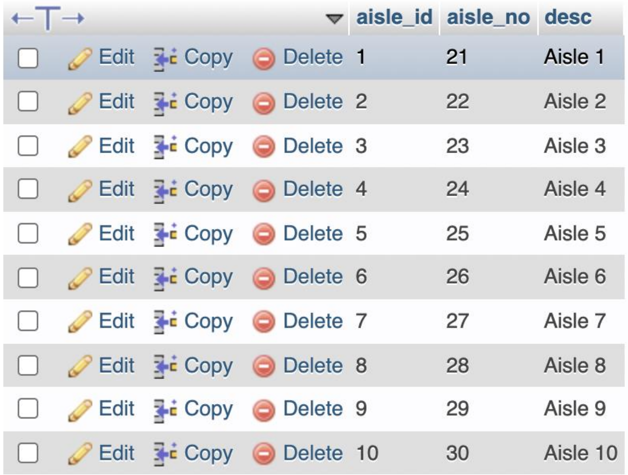
```
INSERT INTO inventories
    (`inventory_id`,`qty`,`in_stock`,`supplier_id`)
VALUES
    (1, 200, 1, 1),
    (2, 160, 1, 2),
    (3, 0, 0, 3),
    (4, 1000, 1, 4),
    (5, 800,1, 5),
    (6, 900,1, 6),
    (7, 650,1, 7),
    (8, 430,1, 8),
    (9, 959,1, 9),
    (10, 642,1, 10);
```



←T→	inventory_id	qty	in_stock 0 for out of stock, 1 for in stock	supplier_id
☐ Edit Copy Delete	1	200	1	1
☐ Edit Copy Delete	2	160	1	2
☐ Edit Copy Delete	3	0	0	3
☐ Edit Copy Delete	4	1000	1	4
☐ Edit Copy Delete	5	800	1	5
☐ Edit Copy Delete	6	900	1	6
☐ Edit Copy Delete	7	650	1	7
☐ Edit Copy Delete	8	430	1	8
☐ Edit Copy Delete	9	959	1	9
☐ Edit Copy Delete	10	642	1	10

Aisles:

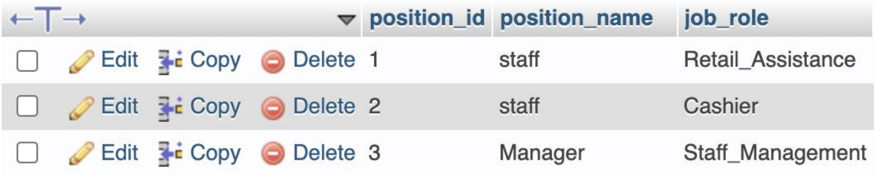
```
INSERT INTO aisles
    (`aisle_id`,`aisle_no`,`desc`)
VALUES
    (1, 21,'Aisle 1'),
    (2, 22,'Aisle 2'),
    (3, 23,'Aisle 3'),
    (4, 24,'Aisle 4'),
    (5, 25,'Aisle 5'),
    (6, 26,'Aisle 6'),
    (7, 27,'Aisle 7'),
    (8, 28,'Aisle 8'),
    (9, 29,'Aisle 9'),

    (10, 30,'Aisle 10');
```



←T→			▼	aisle_id	aisle_no	desc
☐	Edit	Copy	Delete	1	21	Aisle 1
☐	Edit	Copy	Delete	2	22	Aisle 2
☐	Edit	Copy	Delete	3	23	Aisle 3
☐	Edit	Copy	Delete	4	24	Aisle 4
☐	Edit	Copy	Delete	5	25	Aisle 5
☐	Edit	Copy	Delete	6	26	Aisle 6
☐	Edit	Copy	Delete	7	27	Aisle 7
☐	Edit	Copy	Delete	8	28	Aisle 8
☐	Edit	Copy	Delete	9	29	Aisle 9
☐	Edit	Copy	Delete	10	30	Aisle 10

Positions:

```
INSERT INTO `positions` (`position_id`, `position_name`, `job_role`) VALUES

(1, 'staff', 'Retail_Assistance'),

(2, 'staff', 'Cashier'),

(3, 'Manager', 'Staff_Management');
```



←T→			▼	position_id	position_name	job_role
☐	Edit	Copy	Delete	1	staff	Retail_Assistance
☐	Edit	Copy	Delete	2	staff	Cashier
☐	Edit	Copy	Delete	3	Manager	Staff_Management

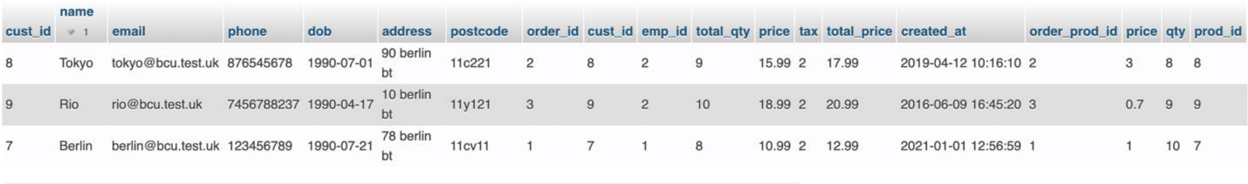
5.3. SQL Queries Testing:

Student (Aditya: 21144695)

1. In SQL queries testing, it is showing three queries to retrieve data from database based on user's need.

Below is the usage of join query, to get all the customers information with order and order product details. Here we are using join which will help us to connect each table with another based on the same customer id and order id with using order by name.

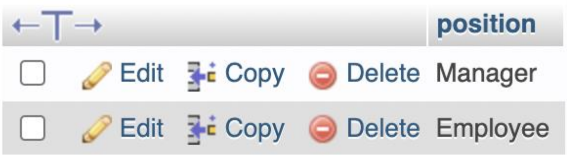
```
select * from customers Inner JOIN orders on customers.cust_id=orders.cust_id Inner JOIN order_products on orders.order_id=order_products.order_id ORDER BY name DESC
```



cust_id	name ▾ 1	email	phone	dob	address	postcode	order_id	cust_id	emp_id	total_qty	price	tax	total_price	created_at	order_prod_id	price	qty	prod_id
8	Tokyo	tokyo@bcu.test.uk	876545678	1990-07-01	90 berlin bt	11c221	2	8	2	9	15.99	2	17.99	2019-04-12 10:16:10	2	3	8	8
9	Rio	rio@bcu.test.uk	7456788237	1990-04-17	10 berlin bt	11y121	3	9	2	10	18.99	2	20.99	2016-06-09 16:45:20	3	0.7	9	9
7	Berlin	berlin@bcu.test.uk	123456789	1990-07-21	78 berlin bt	11cv11	1	7	1	8	10.99	2	12.99	2021-01-01 12:56:59	1	1	10	7

2. Second one is for the usage of distinct. In this example we are using distinct to get all the types of employees position.

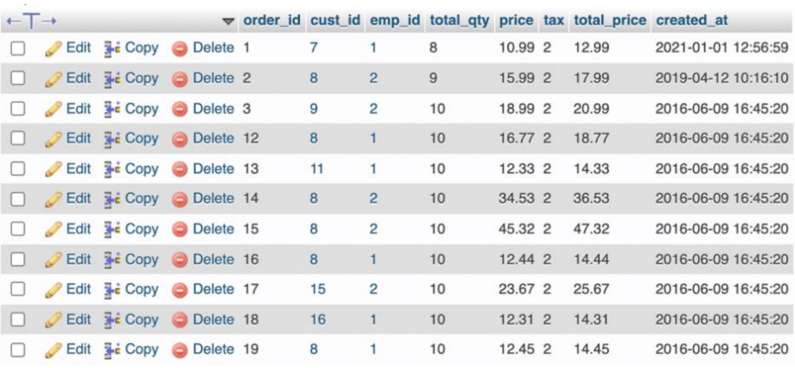
```
select DISTINCT `position` from employees;
```



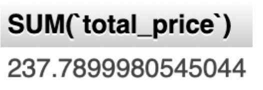
←T→				position
□	Edit	Copy	Delete	Manager
□	Edit	Copy	Delete	Employee

3. Third one is to use one aggregate function. Here we are using sum function to check the total amount of sale from orders table using sum function in mysql query.

```
SELECT SUM(`total_price`) FROM orders;
```



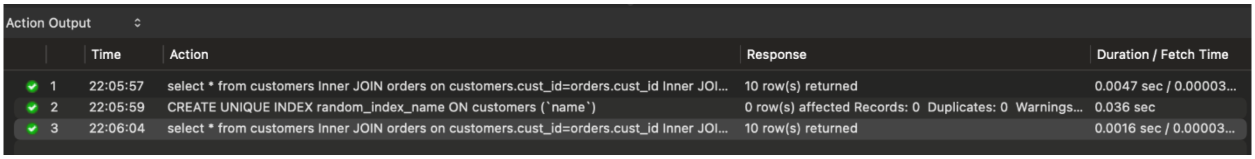
←T→			▾	order_id	cust_id	emp_id	total_qty	price	tax	total_price	created_at
□	Edit	Copy	Delete	1	7	1	8	10.99	2	12.99	2021-01-01 12:56:59
□	Edit	Copy	Delete	2	8	2	9	15.99	2	17.99	2019-04-12 10:16:10
□	Edit	Copy	Delete	3	9	2	10	18.99	2	20.99	2016-06-09 16:45:20
□	Edit	Copy	Delete	12	8	1	10	16.77	2	18.77	2016-06-09 16:45:20
□	Edit	Copy	Delete	13	11	1	10	12.33	2	14.33	2016-06-09 16:45:20
□	Edit	Copy	Delete	14	8	2	10	34.53	2	36.53	2016-06-09 16:45:20
□	Edit	Copy	Delete	15	8	2	10	45.32	2	47.32	2016-06-09 16:45:20
□	Edit	Copy	Delete	16	8	1	10	12.44	2	14.44	2016-06-09 16:45:20
□	Edit	Copy	Delete	17	15	2	10	23.67	2	25.67	2016-06-09 16:45:20
□	Edit	Copy	Delete	18	16	1	10	12.31	2	14.31	2016-06-09 16:45:20
□	Edit	Copy	Delete	19	8	1	10	12.45	2	14.45	2016-06-09 16:45:20



SUM(`total_price`)
237.7899980545044

4. For the query optimization we are using index in our table. Indexing helps for the faster data retrieval. In large database indexing is very helpful to get the fast result.

CREATE UNIQUE INDEX random_index_name ON customers (`name`);



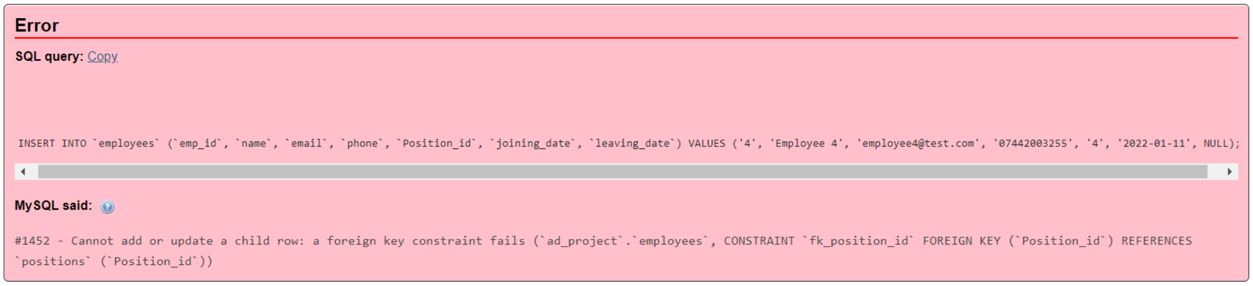
Action Output

		Time	Action	Response	Duration / Fetch Time
✓	1	22:05:57	select * from customers Inner JOIN orders on customers.cust_id=orders.cust_id Inner JOI...	10 row(s) returned	0.0047 sec / 0.00003...
✓	2	22:05:59	CREATE UNIQUE INDEX random_index_name ON customers (`name`)	0 row(s) affected Records: 0 Duplicates: 0 Warnings...	0.036 sec
✓	3	22:06:04	select * from customers Inner JOIN orders on customers.cust_id=orders.cust_id Inner JOI...	10 row(s) returned	0.0016 sec / 0.00003...

In the above image, it is shown that the result we got from query before doing index on table was more (0.0047 sec), compare to the result time duration we got after the indexing (0.0016 sec).

Student (Darshan: 21171488)

5. In this query we will test the foreign key constraint and primary key constraint added in employees table, We tested this constraints with inserting a position_id which is not present in positions table and inserting a duplicate entry but it shows below error



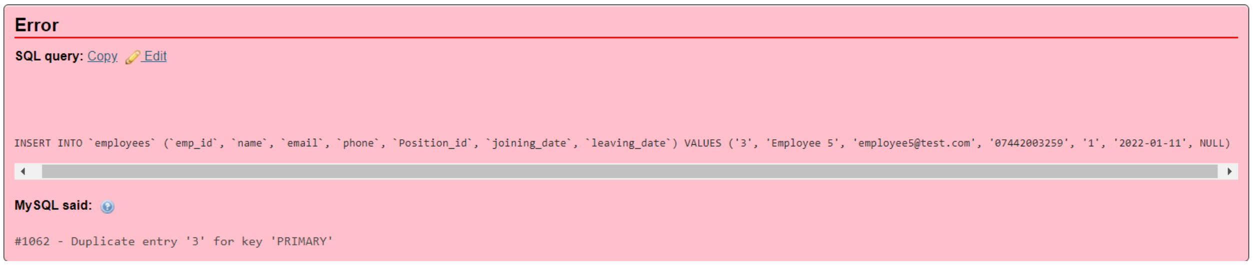
Error

SQL query: Copy

```
INSERT INTO `employees` (`emp_id`, `name`, `email`, `phone`, `Position_id`, `joining_date`, `leaving_date`) VALUES ('4', 'Employee 4', 'employee4@test.com', '07442003255', '4', '2022-01-11', NULL);
```

MySQL said:

```
#1452 - Cannot add or update a child row: a foreign key constraint fails (`ad_project`.`employees`, CONSTRAINT `fk_position_id` FOREIGN KEY (`Position_id`) REFERENCES
`positions` (`Position_id`))
```



Error

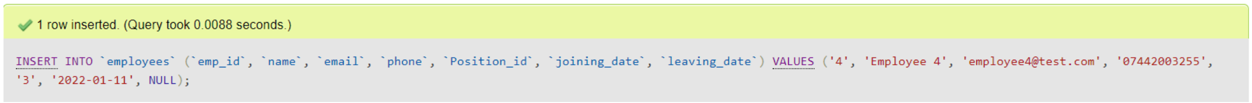
SQL query: Copy Edit

```
INSERT INTO `employees` (`emp_id`, `name`, `email`, `phone`, `Position_id`, `joining_date`, `leaving_date`) VALUES ('3', 'Employee 5', 'employee5@test.com', '07442003259', '1', '2022-01-11', NULL)
```

MySQL said:

```
#1062 - Duplicate entry '3' for key 'PRIMARY'
```

But if we try to insert to insert with correct position_id then it will not show above error.

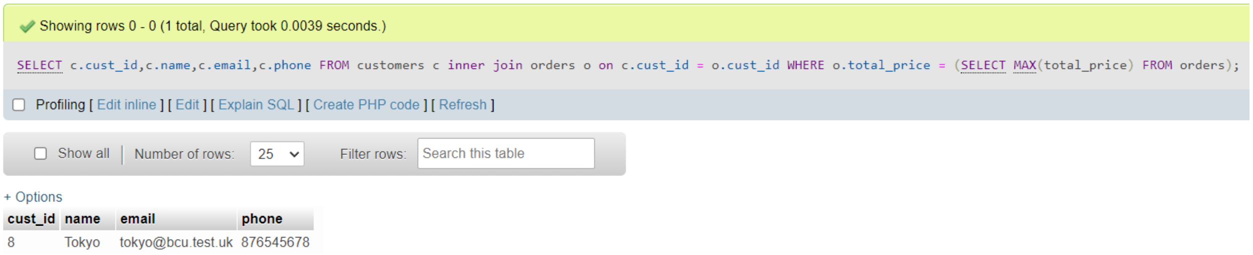


✓ 1 row inserted. (Query took 0.0088 seconds.)

```
INSERT INTO `employees` (`emp_id`, `name`, `email`, `phone`, `Position_id`, `joining_date`, `leaving_date`) VALUES ('4', 'Employee 4', 'employee4@test.com', '07442003255',
'3', '2022-01-11', NULL);
```

6. In this query we will try to fetch customer details who ordered highest amount of order using aggregate function and a subquery.

SELECT c.cust_id,c.name,c.email,c.phone FROM customers c inner join orders o on c.cust_id = o.cust_id WHERE o.total_price = (SELECT MAX(total_price) FROM orders);



✓ Showing rows 0 - 0 (1 total, Query took 0.0039 seconds.)

```
SELECT c.cust_id,c.name,c.email,c.phone FROM customers c inner join orders o on c.cust_id = o.cust_id WHERE o.total_price = (SELECT MAX(total_price) FROM orders);
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
□ Show all
+ Options

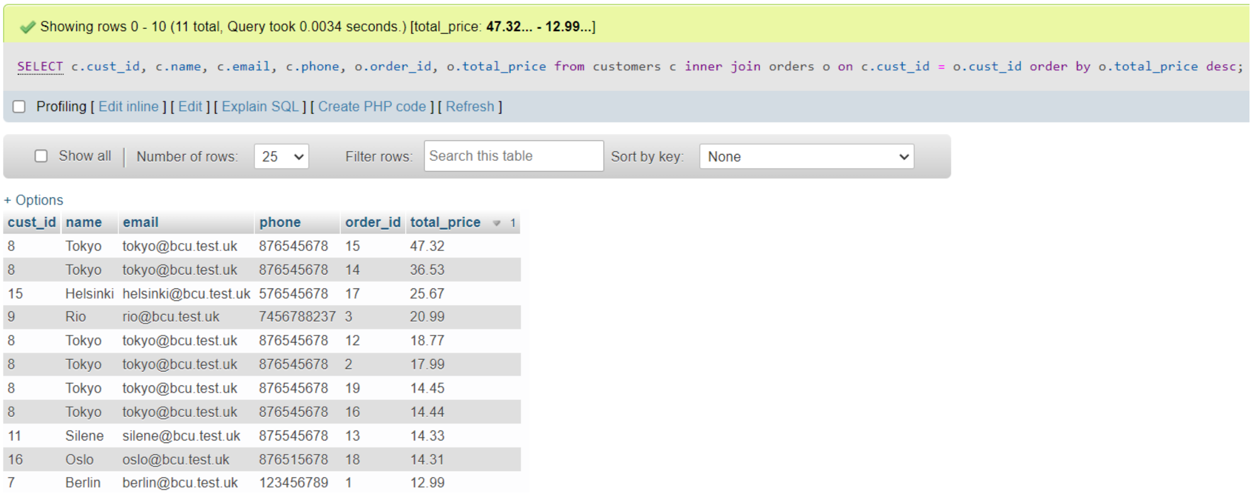
Number of rows: 25

Filter rows: Search this table

cust_id	name	email	phone
8	Tokyo	tokyo@bcu.test.uk	876545678

7. In this query we will try to fetch customers details in descending order of ordered total price.

SELECT c.cust_id, c.name, c.email, c.phone, o.order_id, o.total_price from customers c inner join orders o on c.cust_id = o.cust_id order by o.total_price desc;



✓ Showing rows 0 - 10 (11 total, Query took 0.0034 seconds.) [total_price: **47.32... - 12.99...**]

```
SELECT c.cust_id, c.name, c.email, c.phone, o.order_id, o.total_price from customers c inner join orders o on c.cust_id = o.cust_id order by o.total_price desc;
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
□ Show all
+ Options

Number of rows: 25

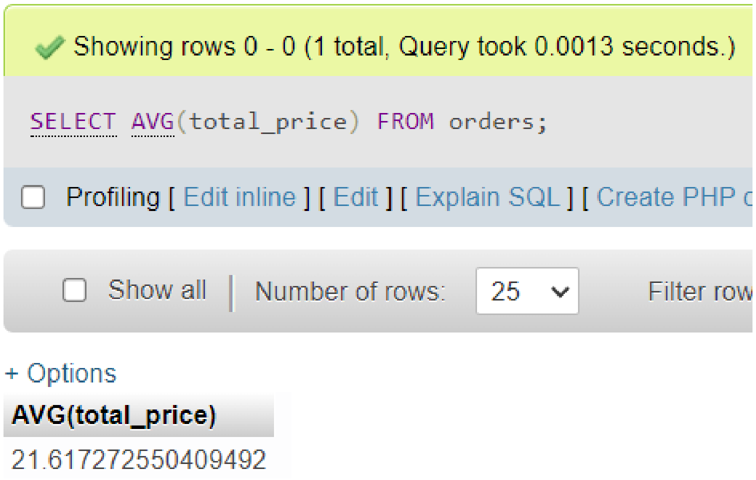
Filter rows: Search this table

Sort by key: None

cust_id	name	email	phone	order_id	total_price ▾ 1
8	Tokyo	tokyo@bcu.test.uk	876545678	15	47.32
8	Tokyo	tokyo@bcu.test.uk	876545678	14	36.53
15	Helsinki	helsinki@bcu.test.uk	576545678	17	25.67
9	Rio	rio@bcu.test.uk	7456788237	3	20.99
8	Tokyo	tokyo@bcu.test.uk	876545678	12	18.77
8	Tokyo	tokyo@bcu.test.uk	876545678	2	17.99
8	Tokyo	tokyo@bcu.test.uk	876545678	19	14.45
8	Tokyo	tokyo@bcu.test.uk	876545678	16	14.44
11	Silene	silene@bcu.test.uk	875545678	13	14.33
16	Oslo	oslo@bcu.test.uk	876515678	18	14.31
7	Berlin	berlin@bcu.test.uk	123456789	1	12.99

8. In this query we will try to average of orders total price.

SELECT AVG(total_price) FROM orders;



✓ Showing rows 0 - 0 (1 total, Query took 0.0013 seconds.)

```
SELECT AVG(total_price) FROM orders;
```

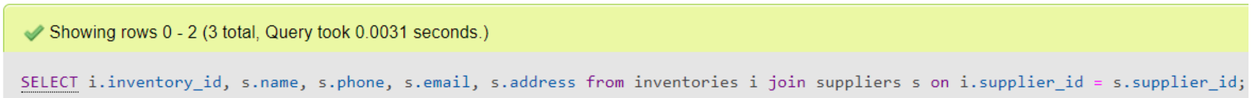
□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP c
□ Show all
Filter row
+ Options

Number of rows: 25

AVG(total_price)
21.617272550409492

9. In this query we will try to average of orders total price.

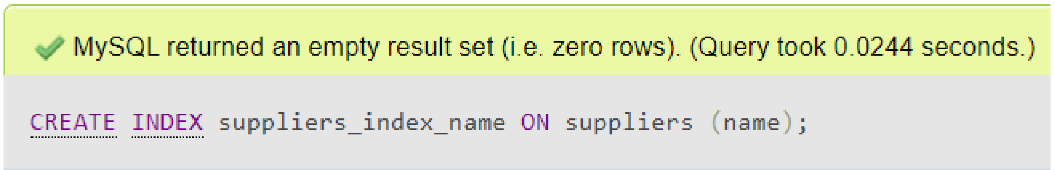
This select query takes around 0.0031 seconds before indexing , so we can minimize this time by adding a index on column



✓ Showing rows 0 - 2 (3 total, Query took 0.0031 seconds.)

```
SELECT i.inventory_id, s.name, s.phone, s.email, s.address from inventories i join suppliers s on i.supplier_id = s.supplier_id;
```

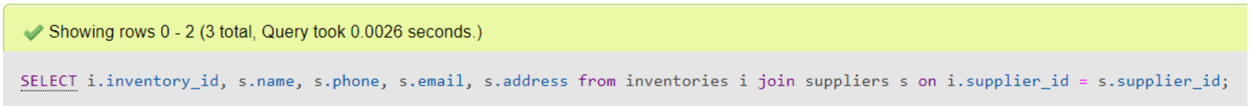
```
SELECT i.inventory_id, s.name, s.phone, s.email, s.address from inventories i
 join suppliers s on i.supplier_id = s.supplier_id;
```



✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0244 seconds.)

```
CREATE INDEX suppliers_index_name ON suppliers (name);
```

```
CREATE INDEX suppliers_index_name ON suppliers (name);
```



✓ Showing rows 0 - 2 (3 total, Query took 0.0026 seconds.)

```
SELECT i.inventory_id, s.name, s.phone, s.email, s.address from inventories i join suppliers s on i.supplier_id = s.supplier_id;
```

After indexing it takes only 0.0026 which is less than before indexing

6 Conclusion:

Our database includes everything that a retail shop may require, and integrating them would help their organization function even more efficiently. since all necessary information is entered into the system that may be retrieved at any moment. Our database maintains data integrity as well as data security.

Our database verifies that all information has been provided, but if any information has been provided incorrectly, our database should display an alert and refuse to save it in the database. We acquired cognitive information of our database prior to establishing it. Moreover, normalization was required to ensure that now the tables are now in the 3rd NF. Ultimately. We stayed on track and feel we created a really well database.