CIS 235 DATABASE PROJECT:

ALDIWAN RESTAURANT

Project Group Number (in Blackboard): **12**
Project Title: ALDIWAN RESTAURANT

Leader	ID	Member Name
✔	2230004065	Razan Abdullah Alqahtani
	2230003587	Alreem Majed Alkhaldi
	2230002699	Norah Abdullah alharkan
	2230000901	Fatima Mohammed Alsayed
	2230000842	Moudhi Mustafa Alhindas
	2230004865	Aljazi edaah alsaiari
	2230000758	Maha Alharthi

Instructor Name: Reem Almualem – Maha Alghamdi
Date of Submission: 30/4/2025

Project Proposal (1 Mark)

Proposal of application:

Aldewan Restaurant System.

Brief Description:

Our project name is "Aldewan Restaurant", it is a restaurant management system based on Java programing with its object-oriented capabilities, to handle regular restaurant objectives and duties. Including placing orders, menu navigation, member accounts, types of billing, all that in a user-friendly graphical interface to make it easy on the members to use it.

Logical Analysis & Business Design:

The restaurant system is designed to manage employees, menu items, orders and takeaway. The manager will be handling all CRUD operation on menu items and personnel. The user will handle all CRUD operation on orders. The system also is capable of handling user errors, to keep the system protected from crashing.

Matching System Functions to the Application Goals:

Functions and Application Properties	Classes Management	Data Input	Professional Appearance	Error Handling	Data Access Tools and Techniques
Log-in: Manager or user	✓	✓	✓	✓	✓
user: takes customer orders, and initiate a bill.	✓	✓	✓		✓
Manager: adds, edits, deletes, items or users.	✓	✓	✓	✓	✓
Input customer data		✓		✓	✓

Entities:

1. **Menu items**
2. **Person**
3. **PayCredit**
4. **Orders**
5. **TakeAway**
6. **Customer**
7. **Inventory**

Systems Users (Actors):

No.	User Role	Privileges
1	User	• Add customer meal requests from the menu. • Edit customer order. • Remove/ cancel items from customer bill, in case of cancelling part of the order. • Cancel the order. • Takes cash from customer. • process the pay method in case of credit card.
2	Manager	• Add/ remove/ edit users. • Add/ remove/ edit menu items. • Add/ remove/ edit inventory items.

ER-Diagram

Requirement 2. Information about each entity-type should be stored with different data types.

1. System components (players):

- User Management
  - Our system will allow the manager to add, edit, and delete users (users, and managers).
  - Each user will have his own password to log in
- Menu Items Management:

  - managers will be able to add, edit, and remove menu items.
  - Each menu item has a name, price, and id.
- Inventory Items Management:
  - Managers follow on the inventory stock.
  - Managers add, view, edit, or delete inventory (CRUD).
  - Managers check the inventory stock limit to request and reorder.
  - Each inventory item has an id, name, description, quantity, unit price, and reorder quantity.
- Order Management:
  - Customers will place their meal orders.
  - Users will be able to add, view, and update orders.
- Payment:
  - Our system will generate bills for orders.
  - It supports multiple payment methods (cash, and credit card).
- Reliability:
  - Our system can be completely relayed on. Because all error checks are taken into consideration, whether system errors or user errors.

2. Entities Attributes of ALDEWAN restaurant: (Primary keys are underlined)

- Person:
  - Attributes: person_id, Name, Role, Password, full_name.
- Menu Items:
  - Attributes: Item_id, Name, Price.
- Inventory Items:
  - Attributes: Inv_id, Name, Description, Unit_price, Reorder_quantity, and person_id.
- Orders: *(Weak Entity)*
  - Attributes: Order_id, user_id, item_id, quantity, total_cost.
- Bill: *(Weak Entity)*
  - Attributes: Bill_id, Order_id, TotalCoast, PaymentMethod.
- Customer:
  - Attributes: Customer_id, Customer_name, Address, Tel *(Multi Value)*.
- TakeAway: *(Weak Entity)*
  - Attributes: Takeaway_id, Customer_id, Order_id, delivery_fees, Customer_name, Tel *(Multi Value)*.
- PayCredit: (*Weak Entity*)

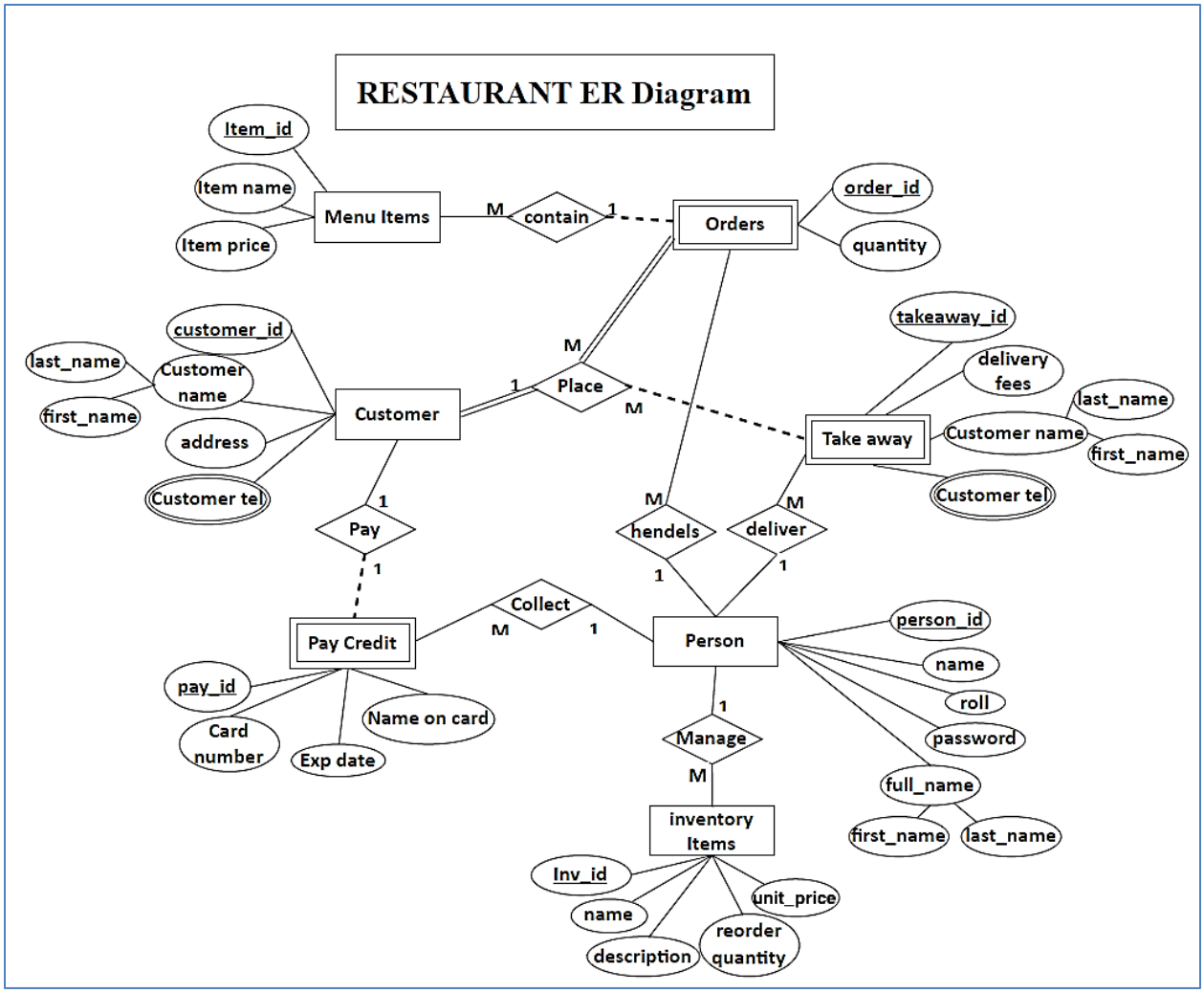
- Attributes: Pay_id, Customer_id, Order_id, name_on_card, exp_date, Card_no.

3. Relationship Between Entities of ALDEWAN restaurant and related functionalities:

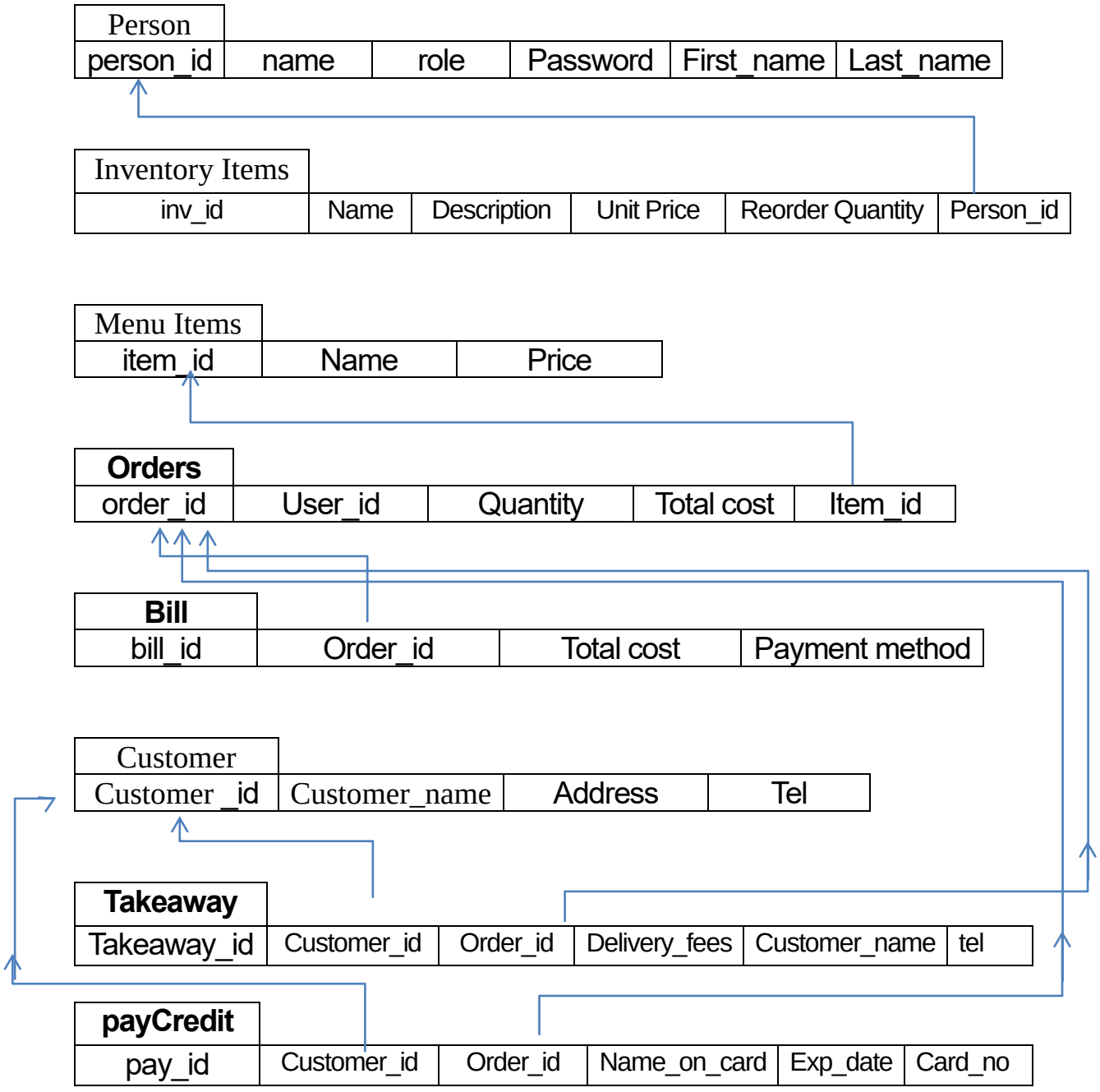
- User - Orders:
  - A user can place multiple orders.
  - Relationship: One-to-Many (1:M).
- Manager – Inventory:
  - A manager role can place inventory orders edit and delete items.
  - Relationship: One-to-Many (1:M).
- Orders - Menu:
  - An order can include multiple menu items.
  - Relationship: One-to-Many(1:M).
- TakeAway – Orders:
  - Takeaway can have many orders.
  - Relationship: One-to-Many.
- Customer – PayCredit:
  - A customer may pay with one credit card.
  - Relationship: One-to-One.

Requirement 3 and 4: The ER diagram fulfills R 3 and R 4.

ALDEWAN RESTAURANT ER Diagram



Tables Mapping of ALDEWAN restaurant:



Tables with full specification of data types for all fields, constraint and Keys:

Table: Person

Attribute name	Description	Datatype	constraint
Person_id	Primary key for this table	Integer	Not null
Name	Login name	String	Not null
Password	Password for person account	String	Not null
Fullname	Complete name for person	String	Not null

Table: Takeaway

Attribute name	Description	Datatype	constraint
Takeaway _id	Primary key for this table	Integer	Not null
Order_id	Foreign key to connect to another table	Integer	Not null
Customer_id	Foreign key to connect to another table	Integer	Not null
delivery_fees	Customer pays extra delivery fees	Double	Not null
Customer_name	Customer name	String	Not null
Customer_address	Customer address, where to deliver the order	String	Not null
Customer_tel	Customer tel, contact	String	Not null
Amount_due	Total amount to pay	Double	Not null

Table: payCredit

Attribute name	Description	Datatype	constraint
pay_id	Primary key for this table	Integer	Not null
Customer_id	Foreign key to connect to another table	Integer	Not null
person_id	Foreign key to connect to another table	Integer	Not null
Cash_witdraw	The amount of money to take from card	Double	Not null
CardNo	Card Number	String	Not null
Name_on_card	Name on card	String	Not null
Exp_date	Expiry date	Date	Not null

Table: Orders

Attribute name	Description	Datatype	constraint
order_id	Primary key for this table	Integer	Not null
user_id	Foreign key to connect to other table	Integer	Not null
item_id	Foreign key to connect to other table	Integer	Not null
Type	Pay type (cash/ Credit, after delivery)	String	Not null
Quantity	Number of items ordered	Integer	Not null
Total_cost	Calculate number * price	Double	Not null

Table: Customer

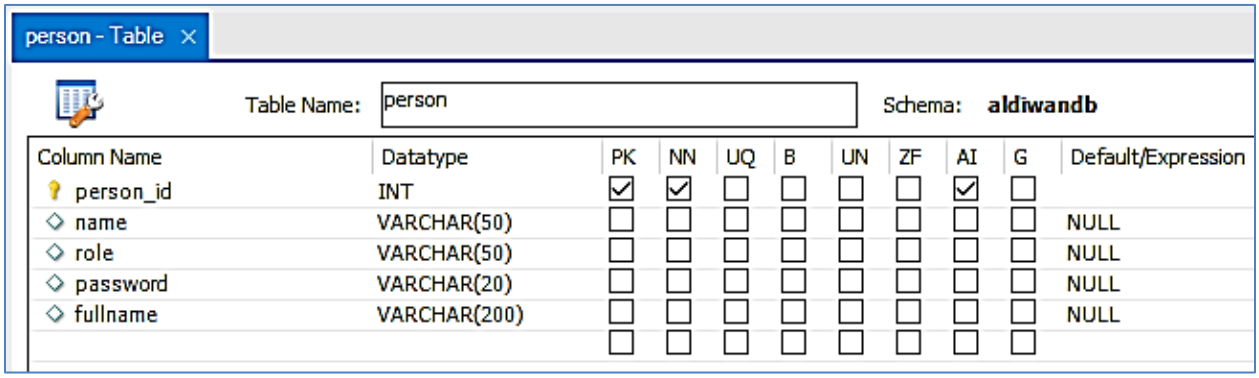
Attribute name	Description	Datatype	constraint
customer_id	Primary key for this table	Integer	Not null
customer _Name	Item name on the menu	String	Not null
Address	Customer address	String	Not null
Tel	Customer telephone	String	not null

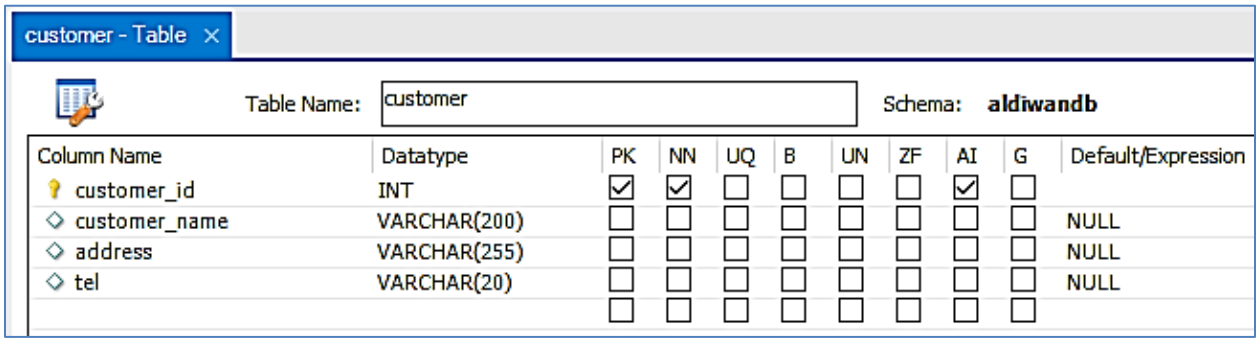
Table: Inventory

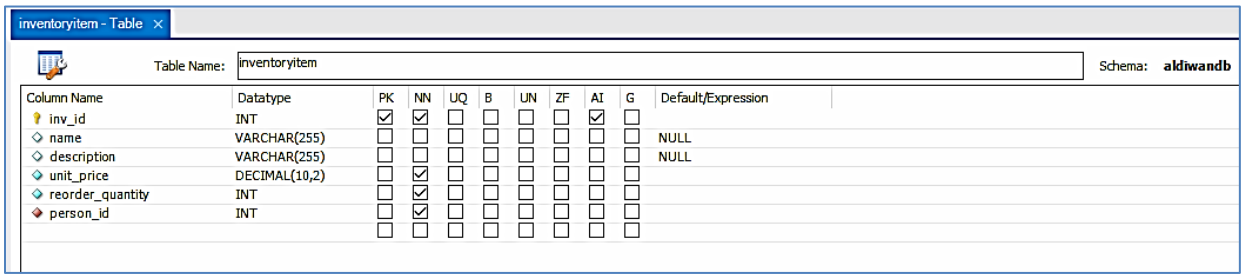
Attribute name	Description	Datatype	constraint
inv_id	Primary key for this table	Integer	Not null
Person_id	Foreign key to connect to other table	Integer	Not null
Name	Item name in stock	String	Not null
Description	Item description	String	Not null
Unit_price	Unit price	Double	Not null
Quantity	Number of items in stock	Double	Not null
Reorder quantity	The minimum limit of stock	Double	Not null

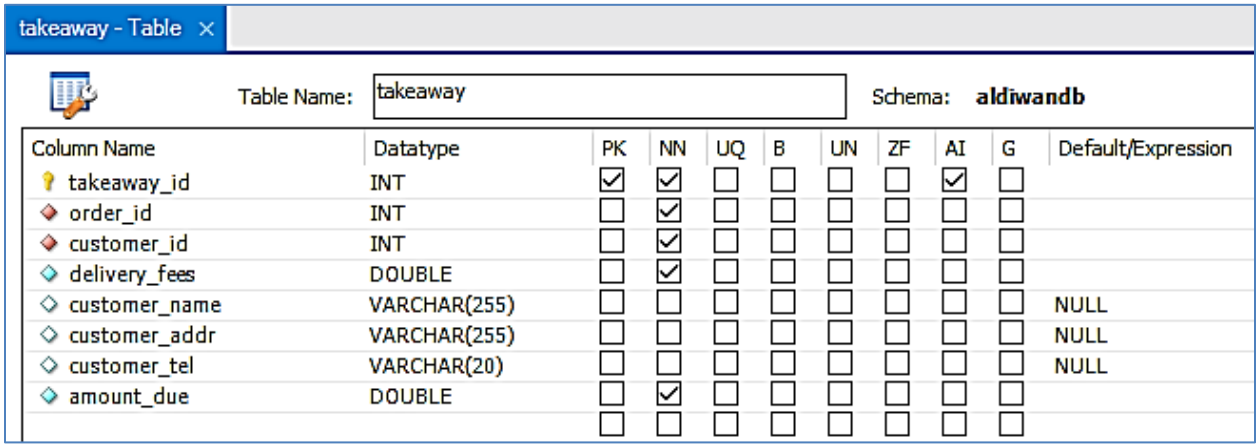
Table: Menu_items

Attribute name	Description	Datatype	constraint
Item_id	Primary key for this table	Integer	Not null
Name	Item name on the menu	String	Not null
Price	Item price	double	Not null





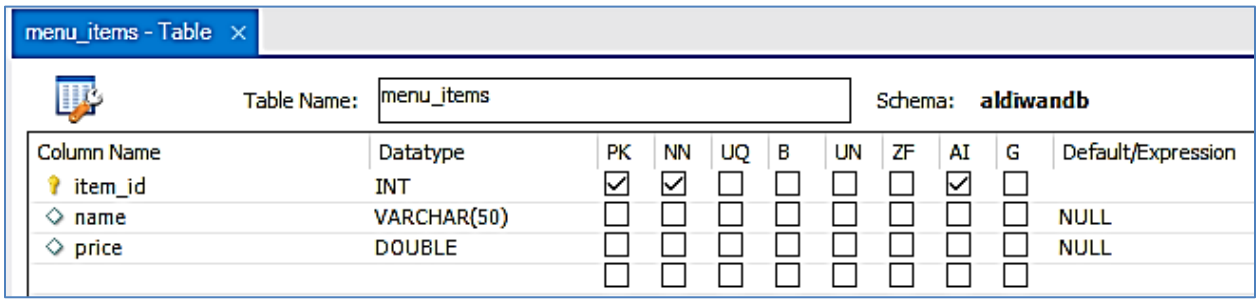




takeaway - Table

Table Name: takeaway Schema: **aldiwandb**

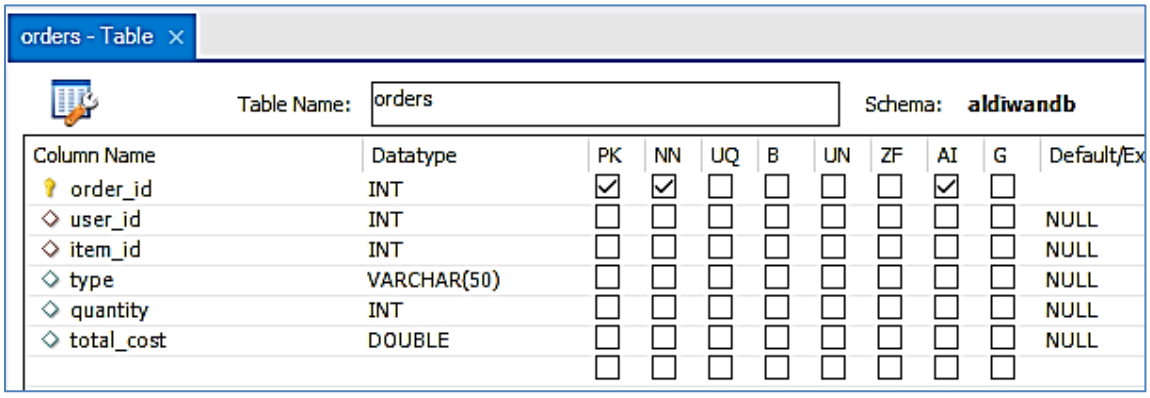
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
takeaway_id	INT	☑	☑	☐	☐	☐	☐	☑	☐	
order_id	INT	☐	☑	☐	☐	☐	☐	☐	☐	
customer_id	INT	☐	☑	☐	☐	☐	☐	☐	☐	
delivery_fees	DOUBLE	☐	☑	☐	☐	☐	☐	☐	☐	
customer_name	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
customer_addr	VARCHAR(255)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
customer_tel	VARCHAR(20)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
amount_due	DOUBLE	☐	☑	☐	☐	☐	☐	☐	☐	



menu_items - Table

Table Name: menu_items Schema: **aldiwandb**

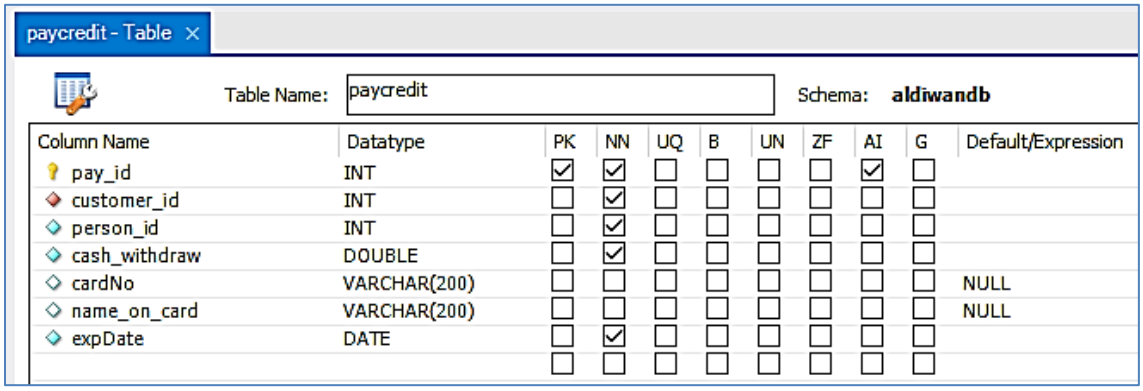
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
item_id	INT	☑	☑	☐	☐	☐	☐	☑	☐	
name	VARCHAR(50)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
price	DOUBLE	☐	☐	☐	☐	☐	☐	☐	☐	NULL



orders - Table

Table Name: orders Schema: **aldiwandb**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Ex
order_id	INT	☑	☑	☐	☐	☐	☐	☑	☐	
user_id	INT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
item_id	INT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
type	VARCHAR(50)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
quantity	INT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
total_cost	DOUBLE	☐	☐	☐	☐	☐	☐	☐	☐	NULL



paycredit - Table

Table Name: paycredit Schema: aldiwandb

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
pay_id	INT	☑	☑	☐	☐	☐	☐	☑	☐	
customer_id	INT	☐	☑	☐	☐	☐	☐	☐	☐	
person_id	INT	☐	☑	☐	☐	☐	☐	☐	☐	
cash_withdraw	DOUBLE	☐	☑	☐	☐	☐	☐	☐	☐	
cardNo	VARCHAR(200)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
name_on_card	VARCHAR(200)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
expDate	DATE	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

Fill Data for all Tables with 10 records: Projects table data

```
4 •  SELECT * FROM inventoryitem;
```

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

inv_id	name	description	unit_price	reorder_quantity
1	Beef Patties	Ground beef patties	2.50	30
2	Burger Buns	Soft sesame buns	0.50	40
3	Lettuce	Fresh iceberg lettuce	0.20	10
4	Tomatoes	Ripe slicing tomatoes	0.30	15
5	Cheese Slices	Cheddar cheese	0.40	20
6	Onions	Red onions	0.15	10
7	Pickles	Dill pickle slices	0.25	20
8	Ketchup	Buckett of ketchup	1.00	5
9	Mayonnaise	Buckett of mayonnaise	1.20	5
10	Fries	Frozen fries	0.10	50
NULL	NULL	NULL	NULL	NULL

```
5 • SELECT * FROM menu_items;
```

Result Grid | Filter Rows: | Edit: | Export/

item_id	name	price
1	Pizza Heart	10
2	Double Cheese Burger	20
3	Burger	10.5
4	Cheesesteak	15.5
5	Cheesesteak Double	18.5
6	Cheesesteak Fire	22.8
7	Cola Bottle 2 Liter	10
8	Cola Bottle 1 Liter	10
9	Cola Can	3
10	Frise	3
11	Pasta Casserole	8.4
12	Pasta Parmesan	10.2
NULL	NULL	NULL

```
4 •  SELECT * FROM orders;
```

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

order_id	user_id	item_id	type	quantity	total_cost
1	1	1	Cash	2	20
2	2	2	Credit	1	10
3	1	3	Takeaway	3	30
4	2	1	Cash	4	40
5	3	4	Takeaway	2	20
6	3	11	Credit	2	22
7	4	2	Takeaway	2	30
8	4	6	Takeaway	2	30
9	5	12	Credit	2	25
10	5	4	Cash	2	22
NULL	NULL	NULL	NULL	NULL	NULL

```
5 •  SELECT * FROM paycredit;
```

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

pay_id	customer_id	person_id	cash_withdraw	cardNo	name_on_card	expDate
1	2	2	50	3423131456	Sara	2027-03-12
2	2	2	40	3423131456	Sara	2027-05-12
3	3	3	44	3423345456	Mohammed	2028-10-01
4	4	3	35	3423567654	Layla	2026-11-10
5	5	6	20	3423131987	Noor	2027-05-10
6	6	6	28	3423131123	Khalid	2028-04-23
7	3	4	45	3423131369	Khalid	2029-12-25
8	3	4	15	3423131687	Abdullah	2026-01-30
9	4	4	27	3423131753	Jaber	2027-05-20
10	4	4	26	3423131487	Al-Mansoori	2028-05-11
11	6	3	35	3423131669	Noor	2028-03-22
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
6 • SELECT * FROM takeaway;
```

takeaway_id	order_id	customer_id	delivery_fees	customer_name	customer_addr	customer_tel	amount_due
21	1	1	5	Ahmed Al-Mansoori	Apt 3, Al-Waha Neigh...	+966 55 123 4567	25
22	2	2	6	Fatima Al-Nasser	Apt 23, King Saud Ro...	+966 50 234 5678	24
23	3	3	7	Abdullah Al-Harbi	Building 14, Prince Fai...	+966 56 345 6789	66
24	4	4	5	Sara Al-Sharif	Villa 8, Al-Raka Distric...	+966 54 456 7890	48
25	5	5	6	Khalid Al-Saud	Villa 11, Al-Murjan Str...	+966 59 567 8901	10
26	6	2	7	Noor Al-Awadhi	Flat 12, Al-Faisaliah St...	+966 58 678 9012	15
27	7	3	5	Omar Al-Farhan	Building 19, Al-Khalidi...	+966 57 789 0123	62
28	8	4	6	Layla Al-Ghamdi	Apartment 5, Al-Nakh...	+966 53 890 1234	24
29	9	5	7	Mohammed Al-Zahrani	House 7, Al-Quds Nei...	+966 52 901 2345	35
30	10	3	5	AishaAl-Jaber	House 7, Al-Quds Nei...	+966 51 012 3456	29

```
5 • SELECT * FROM customer;
```

customer_id	customer_name	address	tel
1	Ahmed Al-Mansoori	Building 14, Prince Fai...	+966 55 123 4567
2	Fatima Al-Nasser	Apt 23, King Saud Ro...	+966 50 234 5678
3	Abdullah Al-Harbi	Villa 8, Al-Raka Distric...	+966 56 345 6789
4	Sara Al-Sharif	Flat 12, Al-Faisaliah St...	+966 54 456 7890
5	Khalid Al-Saud	House 7, Al-Quds Nei...	+966 59 567 8901
6	Noor Al-Awadhi	Building 19, Al-Khalidi...	+966 58 678 9012
7	Omar Al-Farhan	Apartment 5, Al-Nakh...	+966 57 789 0123
8	Layla Al-Ghamdi	Villa 11, Al-Murjan Str...	+966 53 890 1234
9	Mohammed Al-Zahrani	Apt 3, Al-Waha Neigh...	+966 52 901 2345
10	AishaAl-Jaber	Flat 22, Al-Rashid Roa...	+966 51 012 3456

```
3 • SELECT * FROM person;
```

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

person_id	name	role	password	fullname
1	Razan	Manager	rrr	Razan Alqahtani
2	Alreem	user	aaa	Alreem Alkhaldi
3	Norah	user	nnn	Norah Alharkan
4	Sara	user	sss	Sara Mohamed
5	Marwah	user	mmm	Marwah Ahmed
6	Samia	user	sss	Samia Saad
NULL	NULL	NULL	NULL	NULL

Requirement 5 a:

Here in MySQL Workbench, we create the database then the tables, and fill the tables with example data, accordingly we used SQL queries such as CREATE, INSERT, and declared PRIMARY and FOREIGN KEY constraints.

```
-- Database: `aldiwandb`
CREATE DATABASE IF NOT EXISTS `aldiwandb`;
USE `aldiwandb`;

-- Table structure for table `person`
CREATE TABLE `person` (
  `person_id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(50),
  `role` varchar(50),
```

```
  `password` varchar(20),
  `fullname` varchar(200),
  PRIMARY KEY (`person_id`)
);

INSERT INTO `person` VALUES
(1,'Razan','Manager','rrr','Razan Alqahtani'),
(2,'Alreem ','user','aaa','Alreem Alkhaldi'),
(3,'Norah ','user','nnn','Norah Alharkan'),
(4,'Sara','user','sss','Sara Mohamed'),
(5,'Marwah','user','mmm','Marwah Ahmed'),
(6,'Samia','user','sss','Samia Saad'),
(7,'a','Manager','a','Anhar Mohamed');

-- Table structure for table `customer`
CREATE TABLE `customer` (
  `customer_id` int NOT NULL AUTO_INCREMENT,
  `customer_name` varchar(200),
  `address` varchar(255),
  `tel` varchar(20),
  PRIMARY KEY (`customer_id`)
);

INSERT INTO `customer` VALUES
(1,'Ahmed Al-Mansoori','Building 14, Prince Faisal Street,
    Al-Dammam 32421','+966 55 123 4567'),
(2,'Fatima Al-Nasser','Apt 23, King Saud Road, Al-Dammam
    31952','+966 50 234 5678'),
(3,'Abdullah Al-Harbi','Villa 8, Al-Raka District, Al-Dammam
    34231','+966 56 345 6789'),
```

```
(4,'Sara Al-Sharif','Flat 12, Al-Faisaliah Street, Al-Dammam
    31425','+966 54 456 7890'),
(5,'Khalid Al-Saud','House 7, Al-Quds Neighborhood, Al-
    Dammam 32114','+966 59 567 8901'),
(6,'Noor Al-Awadhi','Building 19, Al-Khalidiyah Area, Al-
    Dammam 31967','+966 58 678 9012'),
(7,'Omar Al-Farhan','Apartment 5, Al-Nakheel District, Al-
    Dammam 34321','+966 57 789 0123'),
(8,'Layla Al-Ghamdi','Villa 11, Al-Murjan Street, Al-Dammam
    31987','+966 53 890 1234'),
(9,'Mohammed Al-Zahrani','Apt 3, Al-Waha Neighborhood, Al-
    Dammam 32213','+966 52 901 2345'),
(10,'AishaAl-Jaber','Flat 22, Al-Rashid Road, Al-Dammam
    31456','+966 51 012 3456');

-- Table structure for table `menu_items`
CREATE TABLE `menu_items` (
  `item_id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(50),
  `price` double DEFAULT NULL,
  PRIMARY KEY (`item_id`)
);

-- Dumping data for table `menu_items`
INSERT INTO `menu_items` VALUES
(1,'Pizza Heart',10),
(2,'Double Cheese Burger',20),
(3,'Burger',10.5),
(4,'Cheesesteak',15.5),
(5,'Cheesesteak Double',18.5),
```

```
(6,'Cheesesteak Fire',22.8),
(7,'Cola Bottle 2 Liter',10),
(8,'Cola Bottle 1 Liter',10),
(9,'Cola Can',3),
(10,'Frise',3),
(11,'Pasta Casserole',8.4),
(12,'Pasta Parmesan',10.2);

-- Table structure for table `orders`
CREATE TABLE `orders` (
  `order_id` int NOT NULL AUTO_INCREMENT,
  `user_id` int DEFAULT NULL,
  `item_id` int DEFAULT NULL,
  `type` varchar(50),
  `quantity` int DEFAULT NULL,
  `total_cost` double DEFAULT NULL,
  PRIMARY KEY (`order_id`),
  KEY `user_id` (`user_id`),
  KEY `item_id` (`item_id`),
  CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`item_id`)
REFERENCES `menu_items` (`item_id`) ON DELETE SET DEFAULT,
  CONSTRAINT `orders_ibfk_2` FOREIGN KEY (`user_id`)
REFERENCES `person` (`person_id`) ON DELETE SET DEFAULT

);


-- Dumping data for table `orders`
INSERT INTO `orders` VALUES
(1,1,1,'Cash',2,20),
```

```
(2,2,2,'Credit',1,10),
(3,1,3,'Takeaway',3,30),
(4,2,1,'Cash',4,40),
(5,3,4,'Takeaway',2,20),
(6,3,11,'Credit',2,22),
(7,4,2,'Takeaway',2,30),
(8,4,6,'Takeaway',2,30),
(9,5,12,'Credit',2,25),
(10,5,4,'Cash',2,22);


-- Table structure for table `paycredit`
CREATE TABLE `paycredit` (
  `pay_id` int NOT NULL AUTO_INCREMENT,
  `customer_id` int NOT NULL,
  `person_id` int NOT NULL,
  `cash_withdraw` double NOT NULL,
  `cardNo` varchar(200),
  `name_on_card` varchar(200),
  `expDate` date NOT NULL,
  PRIMARY KEY (`pay_id`),
  KEY `customer_id` (`customer_id`),
  CONSTRAINT `paycredit_ibfk_1` FOREIGN KEY (`customer_id`)
REFERENCES `customer` (`customer_id`) ON DELETE SET DEFAULT
);

-- Dumping data for table `paycredit`
INSERT INTO `paycredit` VALUES
(1,2,2,50,'3423131456','Sara','2027-03-12'),
(2,2,2,40,'3423131456','Sara','2027-05-12'),
```

```
(3,3,3,44,'3423345456','Mohammed','2028-10-01'),
(4,4,3,35,'3423567654','Layla','2026-11-10'),
(5,5,6,20,'3423131987','Noor','2027-05-10'),
(6,6,6,28,'3423131123','Khalid','2028-04-23'),
(7,3,4,45,'3423131369','Khalid','2029-12-25'),
(8,3,4,15,'3423131687','Abdullah','2026-01-30'),
(9,4,4,27,'3423131753','Jaber','2027-05-20'),
(10,4,4,26,'3423131487','Al-Mansoori','2028-05-11'),
(11,6,3,35,'3423131669','Noor','2028-03-22');

-- Table structure for table `takeaway`
CREATE TABLE `takeaway` (
  `takeaway_id` int NOT NULL AUTO_INCREMENT,
  `order_id` int NOT NULL,
  `customer_id` int NOT NULL,
  `delivery_fees` double NOT NULL,
  `customer_name` varchar(255),
  `customer_addr` varchar(255),
  `customer_tel` varchar(20),
  `amount_due` double NOT NULL,
  PRIMARY KEY (`takeaway_id`),
  KEY `order_id` (`order_id`),
  KEY `customer_id` (`customer_id`),
  CONSTRAINT `takeaway_ibfk_1` FOREIGN KEY (`order_id`)
REFERENCES `orders` (`order_id`) ON DELETE SET DEFAULT,
  CONSTRAINT `takeaway_ibfk_2` FOREIGN KEY (`customer_id`)
REFERENCES `customer` (`customer_id`) ON DELETE SET DEFAULT
);

INSERT INTO `takeaway` VALUES
```

```
(21,1,1,5,'Ahmed Al-Mansoori','Apt 3, Al-Waha Neighborhood,
    Al-Dammam 32213','+966 55 123 4567',25),
(22,2,2,6,'Fatima Al-Nasser','Apt 23, King Saud Road, Al-
    Dammam 31952','+966 50 234 5678',24),
(23,3,3,7,'Abdullah Al-Harbi','Building 14, Prince Faisal
    Street, Al-Dammam 32421','+966 56 345 6789',66),
(24,4,4,5,'Sara Al-Sharif','Villa 8, Al-Raka District, Al-
    Dammam 34231','+966 54 456 7890',48),
(25,5,5,6,'Khalid Al-Saud','Villa 11, Al-Murjan Street, Al-
    Dammam 31987','+966 59 567 8901',10),
(26,6,2,7,'Noor Al-Awadhi','Flat 12, Al-Faisaliah Street,
    Al-Dammam 31425','+966 58 678 9012',15),
(27,7,3,5,'Omar Al-Farhan','Building 19, Al-Khalidiyah Area,
    Al-Dammam 31967','+966 57 789 0123',62),
(28,8,4,6,'Layla Al-Ghamdi','Apartment 5, Al-Nakheel
    District, Al-Dammam 34321','+966 53 890 1234',24),
(29,9,5,7,'Mohammed Al-Zahrani','House 7, Al-Quds
    Neighborhood, Al-Dammam 32114','+966 52 901 2345',35),
(30,10,3,5,'AishaAl-Jaber','House 7, Al-Quds Neighborhood,
    Al-Dammam 32114','+966 51 012 3456',29);

-- Table structure for table `inventoryitem`
CREATE TABLE `inventoryitem` (
  `inv_id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(255),
  `description` varchar(255),
  `unit_price` decimal(10,2) NOT NULL,
  `reorder_quantity` int NOT NULL,
  `person_id` int NOT NULL,
  PRIMARY KEY (`inv_id`),
```

```
  KEY `person_id` (`person_id`),
  CONSTRAINT `inventoryitem_ibfk_1` FOREIGN KEY
(`person_id`) REFERENCES `person` (`person_id`) ON DELETE
SET DEFAULT
);

-- Dumping data for table `inventoryitem`
INSERT INTO `inventoryitem` VALUES
(1,'Beef Patties','Ground beef patties',2.50,30,1),
(2,'Burger Buns','Soft sesame buns',0.50,40,1),
(3,'Lettuce','Fresh iceberg lettuce',0.20,10,2),
(4,'Tomatoes','Ripe slicing tomatoes',0.30,15,2),
(5,'Cheese Slices','Cheddar cheese',0.40,20,1),
(6,'Onions','Red onions',0.15,10,2),
(7,'Pickles','Dill pickle slices',0.25,20,1),
(8,'Ketchup','Buckett of ketchup',1.00,5,3),
(9,'Mayonnaise','Buckett of mayonnaise',1.20,5,3),
(10,'Fries','Frozen fries',0.10,50,4);


-- ------------------------------------------------
-- ALL required Queries from Basic to Advanced,
-- ------------------------------------------------
SELECT * FROM takeaway;

DELETE FROM takeaway WHERE order_id =2;

SELECT * FROM takeaway;
```

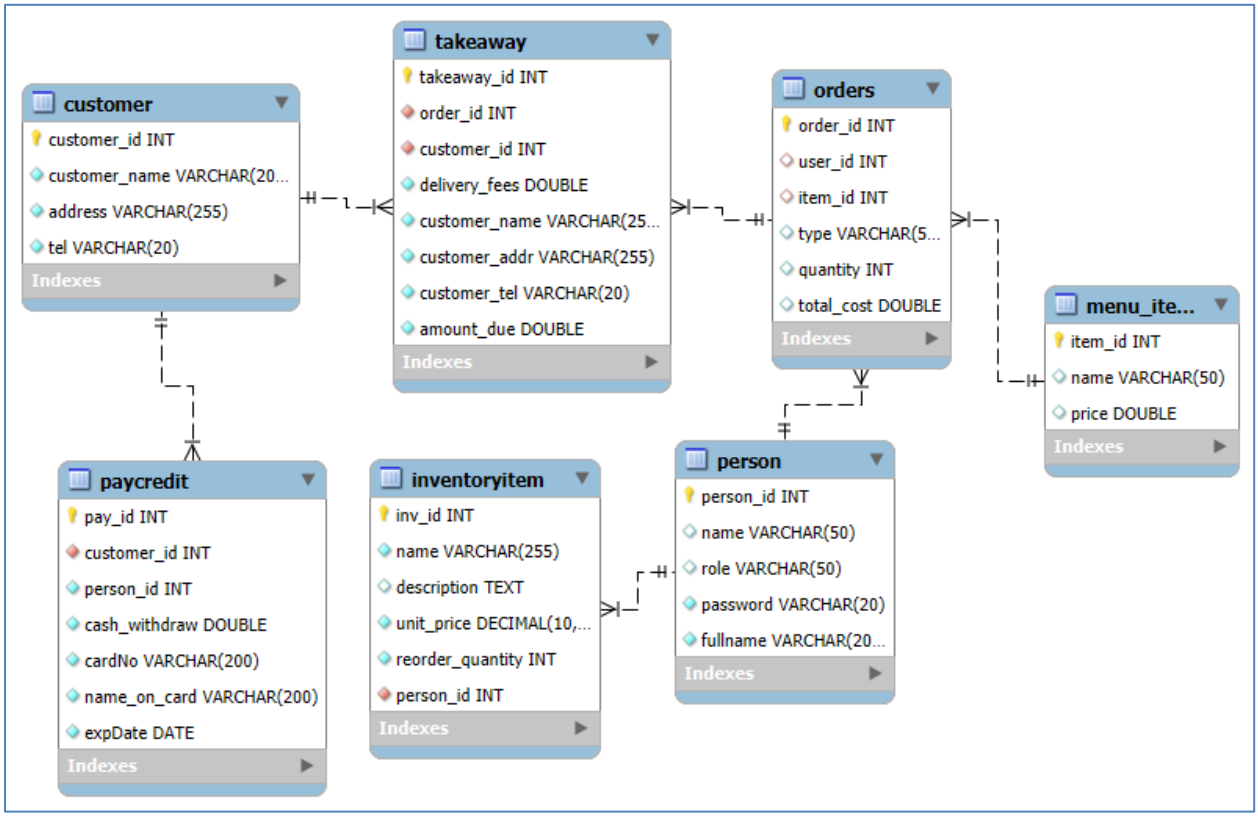
```
UPDATE takeaway SET customer_addr = "Dammam" WHERE
customer_id = 3;

-- ------------------------------------------------
-- >>> Requirement 5 d. ii) 1. Union / Intersect /
    Difference
-- ------------------------------------------------
SELECT * FROM person WHERE name="Razan";
SELECT
    name AS full_name,
    role AS role_or_type
FROM person
UNION
  SELECT
    customer_name AS full_name,
    'Customer' AS role_or_type
  FROM customer;

SELECT p.name AS full_name
FROM person p
INNER JOIN customer c ON p.name = c.customer_name;

-- Find names in `person` but not in
-- `customer` using LEFT JOIN
SELECT p.name AS name
FROM person p
LEFT JOIN customer c ON p.name = c.customer_name
WHERE c.customer_name IS NULL;

-- ------------------------------------------------
```

```
-- >>> Requirent 5 d. ii) 2. Distinct/ aggregate
    functions, In,
-- >>> Between, ISNULL, NOT, LIKE
-- ------------------------------------------------
SELECT DISTINCT role FROM person;

SELECT SUM(total_cost) AS total_revenue
FROM orders;

SELECT
    MIN(total_cost) AS min_order_cost,
    MAX(total_cost) AS max_order_cost
FROM orders;

SELECT person_id, name, role
FROM person
WHERE role IN ('Manager', 'user');

SELECT customer_id, customer_name, tel
FROM customer
WHERE tel BETWEEN '+966 55 000 0000' AND '+966 59 999
    9999';

SELECT order_id, user_id, item_id, type, total_cost
FROM orders
WHERE total_cost BETWEEN 25.00 AND 30.00;

-- ------------------------------------------------
-- >>> Requiremnt 5 d. ii) 3. Nested queries, comparison
    queries
```

```
-- >>> (ALL/ANY), Group By, Order By, Having, INTO, Case
    expression,
-- >>> and compute clause.
-- ------------------------------------------------------


SELECT role, COUNT(*) AS user_count
FROM person
GROUP BY role;


SELECT person_id, name, role
FROM person ORDER BY name ASC;

SELECT customer_id, AVG(delivery_fees) AS
avg_delivery_fee
FROM takeaway
GROUP BY customer_id
HAVING AVG(delivery_fees) > 5.00;

SELECT inv_id, name, SUM(reorder_quantity) AS
total_reorder_quantity
FROM inventoryitem
GROUP BY inv_id, name
HAVING SUM(reorder_quantity) > 15;

SELECT order_id, user_id, item_id, type, total_cost,
    CASE
        WHEN total_cost < 20 THEN 'Low'
        WHEN total_cost BETWEEN 20 AND 30 THEN 'Medium'
```

```
        ELSE 'High'
    END AS cost_category
FROM orders;

-- Procedure with no parameters
DELIMITER //
CREATE PROCEDURE CalculateTotalRevenue()
BEGIN
    SELECT SUM(total_cost) AS total_revenue FROM orders;
END //
DELIMITER ;
CALL CalculateTotalRevenue();

-- Procedure with 2 parameters
DELIMITER //
CREATE PROCEDURE UpdateInventoryItemPrice(
  IN item_id INT, IN new_price DECIMAL(10,2))
BEGIN
    UPDATE inventoryitem
    SET unit_price = new_price
    WHERE inv_id = item_id;
END //
DELIMITER ;
CALL UpdateInventoryItemPrice(1,20);

CREATE VIEW CustomerOrdersSummary AS
  SELECT c.customer_name, o.order_id, o.total_cost
  FROM customer c
  JOIN takeaway t ON c.customer_id = t.customer_id
  JOIN orders o ON t.order_id = o.order_id;
```

```
SELECT * FROM CustomerOrdersSummary;

 CREATE VIEW InventoryReorderList AS
  SELECT name, description, unit_price, reorder_quantity
    FROM inventoryitem
    WHERE reorder_quantity > 0;

SELECT * FROM InventoryReorderList;


-- >>> DDL
-- Create a new table for discounts
CREATE TABLE discounts (
    discount_id INT NOT NULL AUTO_INCREMENT,
    item_id INT NOT NULL,
    discount_percentage DECIMAL(5,2),
    start_date DATE,
    end_date DATE,
    PRIMARY KEY (discount_id),
    FOREIGN KEY (item_id) REFERENCES menu_items(item_id)
);

-- >>> DML
-- this DML is to show the results of the accomplished
    DDL statement above.
-- Insert a discount for menu item with item_id = 1
INSERT INTO discounts (item_id, discount_percentage,
start_date, end_date)
VALUES (1, 15.00, '2023-11-01', '2023-11-30');
SELECT * FROM discounts;
```



Requirement 5 b: Delete tuples: from both base relations (with primary keys) and generated relations (with foreign keys).

Prepare:

```
DELETE FROM takeaway WHERE order_id =2;
```

```
8 •  SELECT * FROM takeaway;
```

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

takeaway_id	order_id	customer_id	delivery_fees	customer_name	customer_addr
21	1	1	5	Ahmed Al-Mansoori	Apt 3, Al-Waha Neigh...
22	2	2	6	Fatima Al-Nasser	Apt 23, King Saud Ro...
23	3	3	7	Abdullah Al-Harbi	Building 14, Prince Fai...
24	4	4	5	Sara Al-Sharif	Villa 8, Al-Raka Distric...

Result:

```
8 • SELECT * FROM takeaway;
9 • DELETE FROM takeaway WHERE order_id =2;
10 • SELECT * FROM takeaway;
```

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

takeaway_id	order_id	customer_id	delivery_fees	customer_name
21	1	1	5	Ahmed Al-Mansoc
23	3	3	7	Abdullah Al-Harbi
24	4	4	5	Sara Al-Sharif
25	5	5	6	Khalid Al-Saud
26	6	2	7	Noor Al-Awadhi

Order ID 2 is deleted from the takeaway table, which contains other foreign keys.

Requirement 5 c: Update tables: for base relations (with primary keys) and generated relations (with foreign keys).

Prepare:

```
UPDATE takeaway SET customer_addr = "Dammam" WHERE
    customer_id = 3;
```

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

takeaway_id	order_id	customer_id	delivery_fees	customer_name	customer_addr
21	1	1	5	Ahmed Al-Mansoori	Apt 3, Al-Waha Neigh...
23	3	3	7	Abdullah Al-Harbi	Building 14, Prince Fai...
24	4	4	5	Sara Al-Sharif	Villa 8, Al-Raka Distric...
25	5	5	6	Khalid Al-Saud	Villa 11, Al-Murjan Str...
26	6	2	7	Noor Al-Awadhi	Flat 12, Al-Faisaliah St...

takeaway 2

Result:

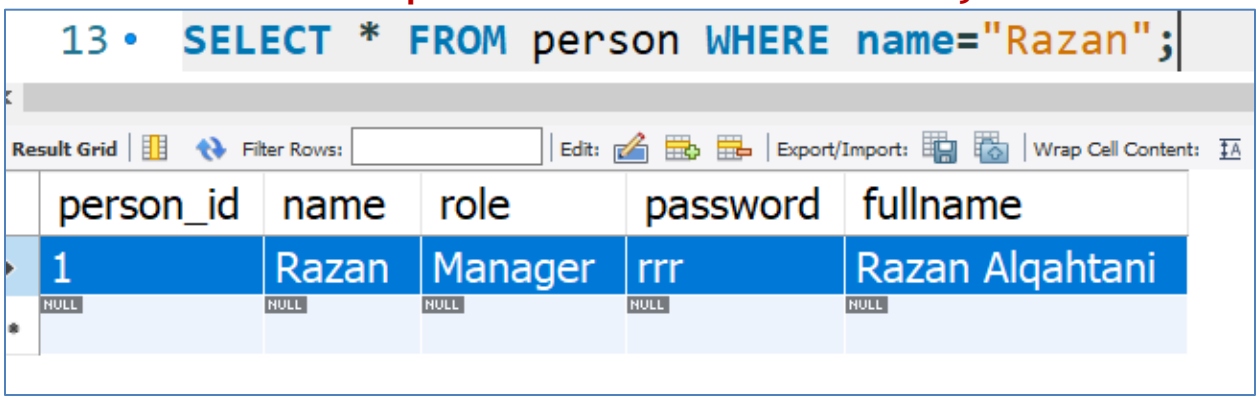
```
10 • SELECT * FROM takeaway;
11 • UPDATE takeaway SET customer_addr = "Dammam" WHERE customer_id = 3;
```

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

takeaway_id	order_id	customer_id	delivery_fees	customer_name	customer_addr
21	1	1	5	Ahmed Al-Mansoori	Apt 3, Al-Waha Neigh...
23	3	3	7	Abdullah Al-Harbi	Dammam
24	4	4	5	Sara Al-Sharif	Villa 8, Al-Raka Distric...
25	5	5	6	Khalid Al-Saud	Villa 11, Al-Murjan Str...

Requirement 5 d. View Advanced Queries Results:
i.) View user's profile (showing user's information)

```
SELECT * FROM person WHERE name="Razan";
```



R5 d. ii) Run at least 15 advanced queries on your application including the following SQL keywords:

R5 d. ii) 1. Union / Intersect / Difference

```
SELECT
    name AS full_name,
    role AS role_or_type
FROM person
UNION
SELECT
    customer_name AS
full_name,
    'Customer' AS
role_or_type
FROM customer;
```

```
14 • SELECT
15       name AS full_name,
16       role AS role_or_type
17   FROM person
18   UNION
19       SELECT
20           customer_name AS full_name,
21           'Customer' AS role_or_type
22       FROM customer;
```

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

full_name	role_or_type
Razan	Manager
Alreem	user
Norah	user
Sara	user
Marwah	user
Samia	user
a	Manager
Ahmed Al-Mansoori	Customer
Fatima Al-Nasser	Customer
Abdullah Al-Harbi	Customer
Sara Al-Sharif	Customer
Khalid Al-Saud	Customer

ult 6 ×

```
-- Since MySQL workbench does not directly support we
    can use a LEFT JOIN
-- Find names in `person` but not in `customer` using
    LEFT JOIN
SELECT p.name AS name
FROM person p
LEFT JOIN customer c ON p.name = c.customer_name
WHERE c.customer_name IS NULL;
```

```
28  -- Find names in `person` but not in
29  -- `customer` using LEFT JOIN
30  SELECT p.name AS name
31  FROM person p
32  LEFT JOIN customer c ON p.name = c.customer_name
33  WHERE c.customer_name IS NULL;
```

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

name
Razan
Alreem
Norah
Sara
Marwah
Samia
a

R5 d. ii) 2. Distinct/ aggregate functions, In, Between, ISNULL, NOT, LIKE

```
SELECT DISTINCT role FROM person;
```

```
35  SELECT DISTINCT role
36  FROM person;
```

Result Grid | Filter Rows: | Export:

role
Manager
user

Aggregate functions (SUM(), MIN(), MAX())

```
SELECT SUM(total_cost) AS total_revenue
FROM orders;
```

```
38 •  SELECT SUM(total_cost) AS total_revenue
39    FROM orders;
```

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

total_revenue
249

```
SELECT
    MIN(total_cost) AS min_order_cost,
    MAX(total_cost) AS max_order_cost
FROM orders;
```

```
41 •  SELECT
42        MIN(total_cost) AS min_order_cost,
43        MAX(total_cost) AS max_order_cost
44    FROM orders;
```

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

min_order_cost	max_order_cost
10	40

```sql
SELECT person_id, name, role
FROM person
WHERE role IN ('Manager', 'user');
```

```
46 • SELECT person_id, name, role
47   FROM person
48   WHERE role IN ('Manager', 'user');
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wra

person_id	name	role
1	Razan	Manager
2	Alreem	user
3	Norah	user
4	Sara	user
5	Marwah	user
6	Samia	user
7	a	Manager
NULL	NULL	NULL

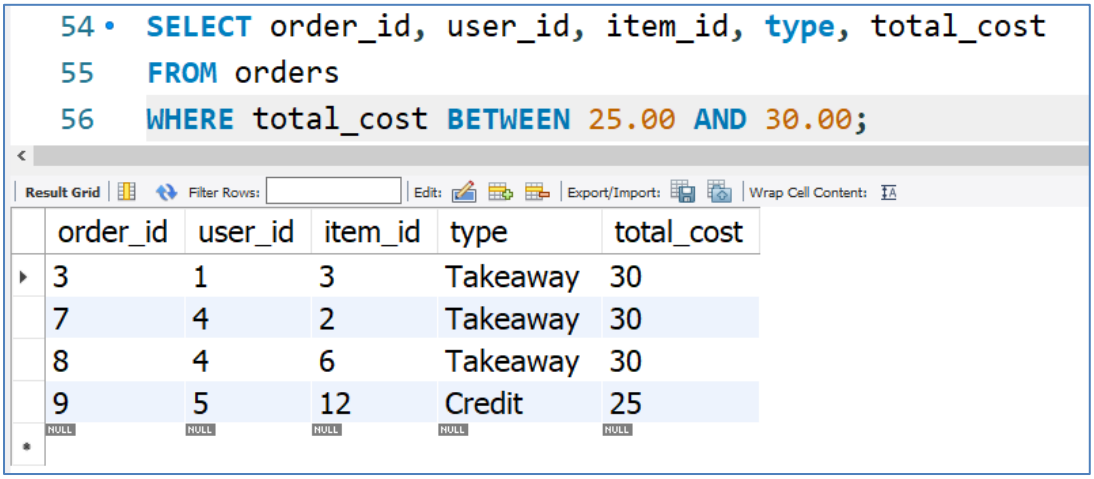
```sql
SELECT customer_id, customer_name, tel
FROM customer
WHERE tel BETWEEN '+966 55 000 0000' AND '+966 59 999 9999';
```

```
50 • SELECT customer_id, customer_name, tel
51   FROM customer
52   WHERE tel BETWEEN '+966 55 000 0000' AND '+966 59 999 9999';
```

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

customer_id	customer_name	tel
1	Ahmed Al-Mansoori	+966 55 123 4567
3	Abdullah Al-Harbi	+966 56 345 6789
5	Khalid Al-Saud	+966 59 567 8901
6	Noor Al-Awadhi	+966 58 678 9012
7	Omar Al-Farhan	+966 57 789 0123
NULL	NULL	NULL

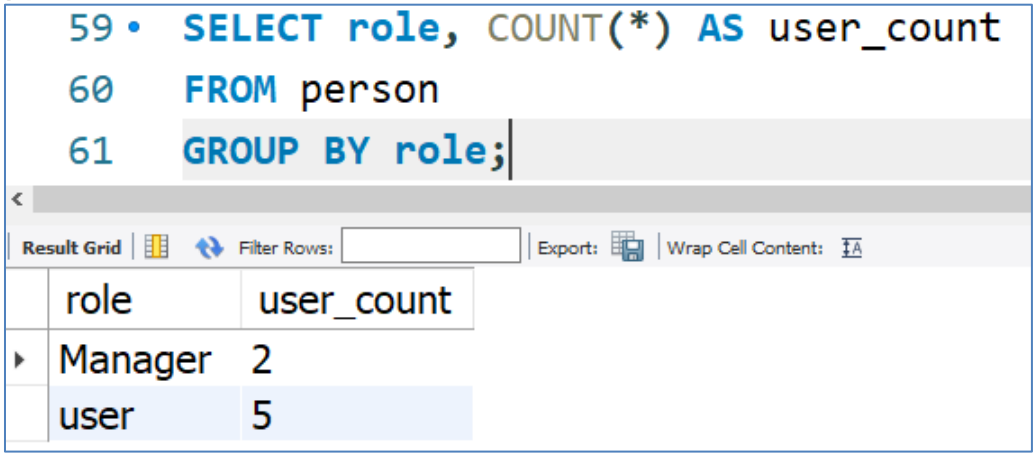
```
SELECT order_id, user_id, item_id, type, total_cost
FROM orders
WHERE total_cost BETWEEN 25.00 AND 30.00;
```



order_id	user_id	item_id	type	total_cost
3	1	3	Takeaway	30
7	4	2	Takeaway	30
8	4	6	Takeaway	30
9	5	12	Credit	25
NULL	NULL	NULL	NULL	NULL

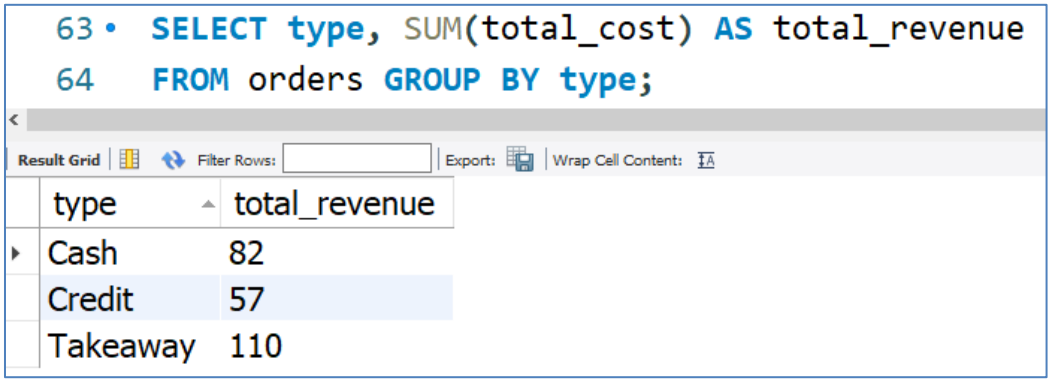
R5 d. ii) 3. Nested queries, comparison queries (ALL/ANY), Group By, Order By, Having, INTO, Case expression, and compute clause.

```
SELECT role, COUNT(*) AS user_count
FROM person GROUP BY role;
```

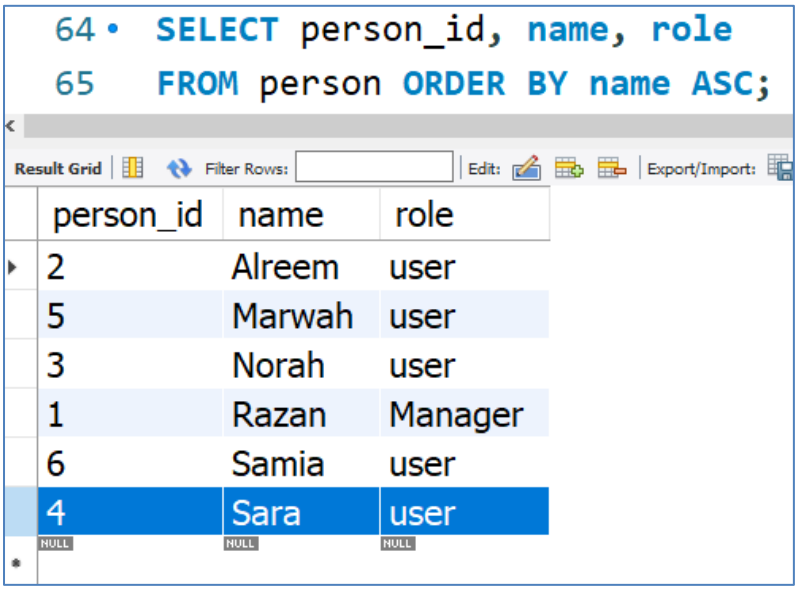


role	user_count
Manager	2
user	5

```
SELECT type, SUM(total_cost) AS total_revenue
FROM orders GROUP BY type;
```




```
SELECT person_id, name, role
FROM person ORDER BY name ASC;
```



```sql
SELECT customer_id, AVG(delivery_fees) AS avg_delivery_fee
FROM takeaway
GROUP BY customer_id
HAVING AVG(delivery_fees) > 5.00;
```

```
67 • SELECT customer_id, AVG(delivery_fees) AS avg_delivery_fee
68   FROM takeaway
69   GROUP BY customer_id
70   HAVING AVG(delivery_fees) > 5.00;
```

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

customer_id	avg_delivery_fee
2	6.5
3	5.666666666666667
4	5.5
5	6.5

```sql
SELECT inv_id, name, SUM(reorder_quantity) AS total_reorder_quantity
FROM inventoryitem
GROUP BY inv_id, name
HAVING SUM(reorder_quantity) > 15;
```

```
72 • SELECT inv_id, name, SUM(reorder_quantity) AS total_reorder_quantity
73   FROM inventoryitem
74   GROUP BY inv_id, name
75   HAVING SUM(reorder_quantity) > 15;
```

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

inv_id	name	total_reorder_quantity
1	Beef Patties	30
2	Burger Buns	40
5	Cheese Slices	20
7	Pickles	20
10	Fries	50

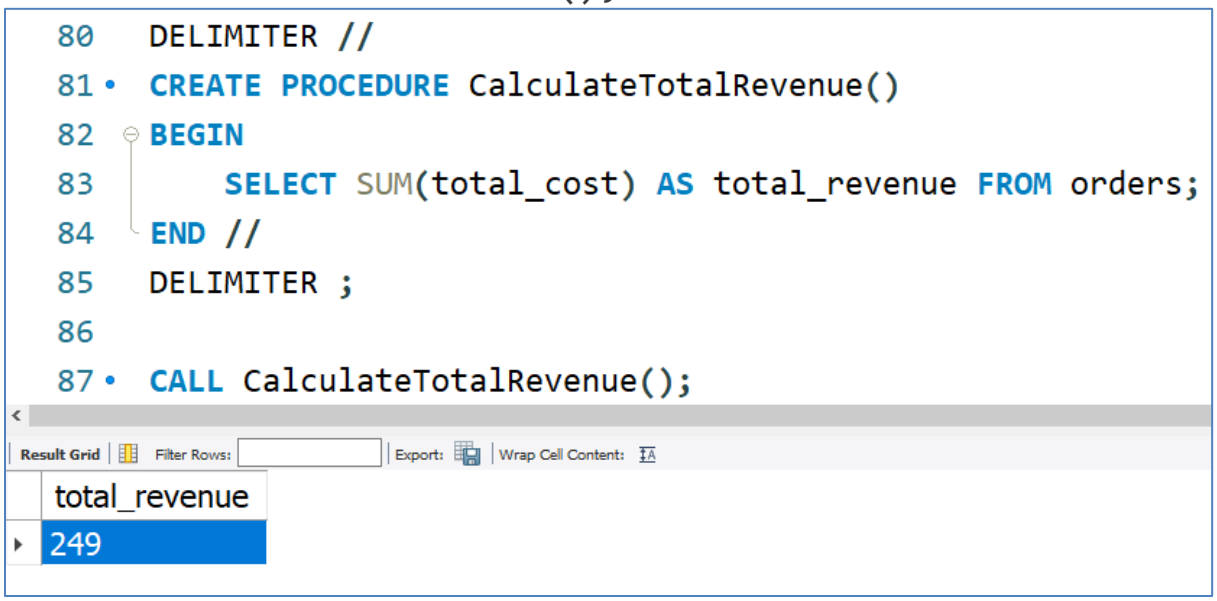
```sql
SELECT order_id, user_id, item_id, type, total_cost,
    CASE
        WHEN total_cost < 20 THEN 'Low'
        WHEN total_cost BETWEEN 20 AND 30 THEN 'Medium'
        ELSE 'High'
    END AS cost_category
FROM orders;
```

```sql
77 • SELECT order_id, user_id, item_id, type, total_cost,
78       CASE
79           WHEN total_cost < 20 THEN 'Low'
80           WHEN total_cost BETWEEN 20 AND 30 THEN 'Medium'
81           ELSE 'High'
82       END AS cost_category
83   FROM orders;
```

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

order_id	user_id	item_id	type	total_cost	cost_category
1	1	1	Cash	20	Medium
2	2	2	Credit	10	Low
3	1	3	Takeaway	30	Medium
4	2	1	Cash	40	High
5	3	4	Takeaway	20	Medium
6	3	11	Credit	22	Medium
7	4	2	Takeaway	30	Medium
8	4	6	Takeaway	30	Medium
9	5	12	Credit	25	Medium
10	5	4	Cash	22	Medium

```
-- Now we will create some views and stored procedures,
-- to fulfill this:
-- b. Create proper stored procedures, views, and
     discussed related functionalities
-- and:
-- You must ensure the application offers the following
     aspects of DB functionalities:
-- stored procedures, views...
DELIMITER //
CREATE PROCEDURE CalculateTotalRevenue()
BEGIN
    SELECT SUM(total_cost) AS total_revenue FROM orders;
END //
DELIMITER ;

CALL CalculateTotalRevenue();
```



```
DELIMITER //
CREATE PROCEDURE UpdateInventoryItemPrice(
  IN item_id INT, IN new_price DECIMAL(10,2))
BEGIN
    UPDATE inventoryitem
    SET unit_price = new_price
    WHERE inv_id = item_id;
END //
DELIMITER ;

CALL UpdateInventoryItemPrice(1,3.50);
SELECT * FROM inventoryitem;
```

```
4  CREATE PROCEDURE UpdateInventoryItemPrice(
5    IN item_id INT, IN new_price DECIMAL(10,2))
6  BEGIN
7      UPDATE inventoryitem
8      SET unit_price = new_price
9      WHERE inv_id = item_id;
10 END //
11 DELIMITER ;
12 CALL UpdateInventoryItemPrice(1,3.50);
```

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

inv_id	name	description	unit_price	reorde
1	Beef Patties	Ground beef patties	3.50	30

```sql
CREATE VIEW CustomerOrdersSummary AS
SELECT
    c.customer_name,
    o.order_id,
    o.total_cost
FROM
    customer c
JOIN
    takeaway t ON c.customer_id = t.customer_id
JOIN
    orders o ON t.order_id = o.order_id;
```

```sql
3 • SELECT * FROM aldiwandb.customerorderssummary;
```

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

customer_name	order_id	total_cost
Ahmed Al-Mansoori	1	20
Fatima Al-Nasser	2	10
Fatima Al-Nasser	6	22
Abdullah Al-Harbi	3	30
Abdullah Al-Harbi	7	30
Abdullah Al-Harbi	10	22
Sara Al-Sharif	4	40
Sara Al-Sharif	8	30
Khalid Al-Saud	5	20
Khalid Al-Saud	9	25

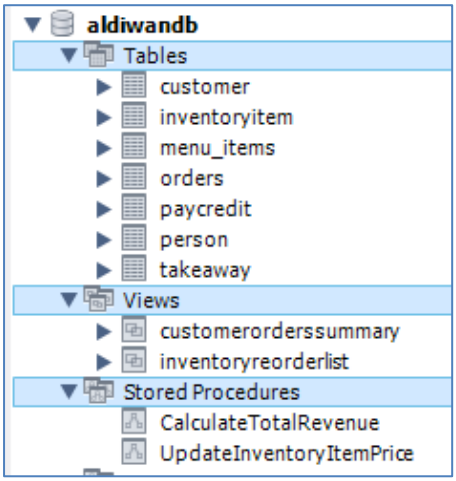
```
CREATE VIEW InventoryReorderList AS
  SELECT name, description, unit_price, reorder_quantity
    FROM inventoryitem
    WHERE reorder_quantity > 0;

SELECT * FROM InventoryReorderList;
```

```
 99 •  CREATE VIEW InventoryReorderList AS
100       SELECT name, description, unit_price, reorder_quantity
101           FROM inventoryitem
102           WHERE reorder_quantity > 0;
103
104 •  SELECT * FROM InventoryReorderList;
```

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

name	description	unit_price	reorder_quantity
Beef Patties	Ground beef patties	3.50	30
Burger Buns	Soft sesame buns	0.50	40
Lettuce	Fresh iceberg lettuce	0.20	10
Tomatoes	Ripe slicing tomatoes	0.30	15
Cheese Slices	Cheddar cheese	0.40	20

Here the MySQL workbench window shows the database structure.



R5 d. iii.) Use DML (Data Manipulation Language) for (insert/update/delete). And at least one DDL (Data Definition Language) such as Create Table.

- **DML**

```
-- Insert a new customer
INSERT INTO customer (customer_name, address,
tel)VALUES
('Youssef Al-Sayed', 'Building 25, Al-Nakheel
District, Al-Dammam 34321', '+966 55 987 6543');

-- Update the phone number of a specific customer
UPDATE customer
SET tel = '+966 50 111 2222'
WHERE customer_id = 1;

-- Delete an order with order_id = 1
DELETE FROM orders
WHERE order_id = 1;
```

- **DDL**

```
-- Create a new table for discounts
CREATE TABLE discounts (
    discount_id INT NOT NULL AUTO_INCREMENT,
    item_id INT NOT NULL,
    discount_percentage DECIMAL(5,2),
    start_date DATE,
    end_date DATE,
    PRIMARY KEY (discount_id),
```

```
    FOREIGN KEY (item_id) REFERENCES menu_items(item_id)
);
-- to make sure the DDL worked we need to use DML to
    check that we are on the right path.
-- Insert a discount 15% for menu item with item_id=1
INSERT INTO discounts (item_id, discount_percentage,
    start_date, end_date)
VALUES (1, 15.00, '2023-11-01', '2023-11-30');
SELECT * FROM discounts;
```

```
109 • CREATE TABLE discounts (
110     discount_id INT NOT NULL AUTO_INCREMENT,
111     item_id INT NOT NULL,
112     discount_percentage DECIMAL(5,2),
113     start_date DATE,
114     end_date DATE,
115     PRIMARY KEY (discount_id),
116     FOREIGN KEY (item_id) REFERENCES menu_items(item_id)
117 );
118  -- Insert a discount for menu item with item_id = 1
119 • INSERT INTO discounts (item_id, discount_percentage, start_date, end_date)
120  VALUES (1, 15.00, '2023-11-01', '2023-11-30');
121 • SELECT * FROM discounts;
```

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

discount_id	item_id	discount_percentage	start_date	end_date
1	1	15.00	2023-11-01	2023-11-30
NULL	NULL	NULL	NULL	NULL