



## Assignment-Courier Management System

### Instructions

- Project submissions should be done through the participants' Github repository and the link should be shared with trainers and Hexavarsity.
- Each section builds upon the previous one, and by the end, you will have a **Courier Management System** implemented with a strong focus on SQL, control flow statements, loops, arrays, collections, exception handling, database interaction and Unit Testing.
- Follow object-oriented principles throughout the project. Use classes and objects to model real-world entities, encapsulate data and behavior, and ensure code reusability.
- Throw user defined exceptions from corresponding methods and handled.
- The following Directory structure is to be followed in the application.
  - **entity**
    - Create entity classes in this package. All entity class should not have any business logic.
  - **dao**
    - Create Service Provider interface to showcase functionalities.
    - Create the implementation class for the above interface with db interaction.
  - **exception**
    - Create user defined exceptions in this package and handle exceptions whenever needed.
  - **util**
    - Create a DBPropertyUtil class with a static function which takes property file name as parameter and returns connection string.
    - Create a DBConnUtil class which holds static method which takes connection string as parameter file and returns connection object(Use method defined in DBPropertyUtil class to get the connection String).
  - **main**
    - Create a class MainModule and demonstrate the functionalities in a menu driven application.

### Task1 Database Design

Design a SQL schema for a Courier Management System with tables for Customers, Couriers, Orders, and Parcels. Define the relationships between these tables using appropriate foreign keys.

#### Requirements:

- Define the Database Schema • Create SQL tables for entities such as **User, Courier, Employee, Location, Payment**
- Define relationships between these tables (**one-to-many, many-to-many, etc.**).
- **Populate Sample Data** • Insert sample data into the tables to simulate real-world scenarios.

#### User Table:

##### User

(UserID INT PRIMARY KEY,  
Name VARCHAR(255),  
Email VARCHAR(255) UNIQUE,



Password VARCHAR(255),  
ContactNumber VARCHAR(20),  
Address TEXT  
);

**Courier**

(CourierID INT PRIMARY KEY,  
SenderName VARCHAR(255),  
SenderAddress TEXT,  
ReceiverName VARCHAR(255),  
ReceiverAddress TEXT,  
Weight DECIMAL(5, 2),  
Status VARCHAR(50),  
TrackingNumber VARCHAR(20) UNIQUE,  
DeliveryDate DATE);

**CourierServices**

(ServiceID INT PRIMARY KEY,  
ServiceName VARCHAR(100),  
Cost DECIMAL(8, 2));

**Employee Table:**

(EmployeeID INT PRIMARY KEY,  
Name VARCHAR(255),  
Email VARCHAR(255) UNIQUE,  
ContactNumber VARCHAR(20),  
Role VARCHAR(50),  
Salary DECIMAL(10, 2));

**Location Table:**

(LocationID INT PRIMARY KEY,  
LocationName VARCHAR(100),  
Address TEXT);

**Payment Table:**

(PaymentID INT PRIMARY KEY,  
CourierID INT,  
LocationId INT,  
Amount DECIMAL(10, 2),  
PaymentDate DATE,  
FOREIGN KEY (CourierID) REFERENCES Couriers(CourierID),  
FOREIGN KEY (LocationID) REFERENCES Location(LocationID));

**Task 2: Select,Where**

Solve the following queries in the Schema that you have created above

1. List all customers:
2. List all orders for a specific customer:
3. List all couriers:
4. List all packages for a specific order:
5. List all deliveries for a specific courier:
6. List all undelivered packages:
7. List all packages that are scheduled for delivery today:
8. List all packages with a specific status:
9. Calculate the total number of packages for each courier.
10. Find the average delivery time for each courier
11. List all packages with a specific weight range:
12. Retrieve employees whose names contain 'John'
13. Retrieve all courier records with payments greater than \$50.

**Task 3: GroupBy, Aggregate Functions, Having, Order By, where**

14. Find the total number of couriers handled by each employee.
15. Calculate the total revenue generated by each location
16. Find the total number of couriers delivered to each location.
17. Find the courier with the highest average delivery time:
18. Find Locations with Total Payments Less Than a Certain Amount
19. Calculate Total Payments per Location
20. Retrieve couriers who have received payments totaling more than \$1000 in a specific location (LocationID = X):
21. Retrieve couriers who have received payments totaling more than \$1000 after a certain date (PaymentDate > 'YYYY-MM-DD'):
22. Retrieve locations where the total amount received is more than \$5000 before a certain date (PaymentDate > 'YYYY-MM-DD')

**Task 4: Inner Join,Full Outer Join, Cross Join, Left Outer Join,Right Outer Join**

23. Retrieve Payments with Courier Information
24. Retrieve Payments with Location Information
25. Retrieve Payments with Courier and Location Information
26. List all payments with courier details
27. Total payments received for each courier
28. List payments made on a specific date



29. Get Courier Information for Each Payment
30. Get Payment Details with Location
31. Calculating Total Payments for Each Courier
32. List Payments Within a Date Range
33. Retrieve a list of all users and their corresponding courier records, including cases where there are no matches on either side
34. Retrieve a list of all couriers and their corresponding services, including cases where there are no matches on either side
35. Retrieve a list of all employees and their corresponding payments, including cases where there are no matches on either side
36. List all users and all courier services, showing all possible combinations.
37. List all employees and all locations, showing all possible combinations:
38. Retrieve a list of couriers and their corresponding sender information (if available)
39. Retrieve a list of couriers and their corresponding receiver information (if available):
40. Retrieve a list of couriers along with the courier service details (if available):
41. Retrieve a list of employees and the number of couriers assigned to each employee:
42. Retrieve a list of locations and the total payment amount received at each location:
43. Retrieve all couriers sent by the same sender (based on SenderName).
44. List all employees who share the same role.
45. Retrieve all payments made for couriers sent from the same location.
46. Retrieve all couriers sent from the same location (based on SenderAddress).
47. List employees and the number of couriers they have delivered:
48. Find couriers that were paid an amount greater than the cost of their respective courier services

**Scope: Inner Queries, Non Equi Joins, Equi joins, Exist, Any, All**

49. Find couriers that have a weight greater than the average weight of all couriers
50. Find the names of all employees who have a salary greater than the average salary:
51. Find the total cost of all courier services where the cost is less than the maximum cost
52. Find all couriers that have been paid for
53. Find the locations where the maximum payment amount was made
54. Find all couriers whose weight is greater than the weight of all couriers sent by a specific sender (e.g., 'SenderName'):

**Coding**

**Task 1: Control Flow Statements**

1. Write a program that checks whether a given order is delivered or not based on its status (e.g., "Processing," "Delivered," "Cancelled"). Use if-else statements for this.



2. Implement a **switch-case statement** to categorize parcels based on their weight into "Light," "Medium," or "Heavy."

3. Implement User Authentication 1. Create a login system for employees and customers using Java **control flow statements**.

4. Implement Courier Assignment Logic 1. Develop a mechanism to assign couriers to shipments based on predefined criteria (e.g., **proximity, load capacity**) using loops.

### **Task 2: Loops and Iteration**

5. Write a Java program that uses a for loop to display all the orders for a specific customer.

6. Implement a **while loop** to track the real-time location of a courier until it reaches its destination.

### **Task 3: Arrays and Data Structures**

7. Create an array to store the tracking history of a parcel, where each entry represents a location update.

8. Implement a method to find the nearest available courier for a new order using an **array of couriers**.

### **Task 4: Strings, 2d Arrays, user defined functions, Hashmap**

9. **Parcel Tracking:** Create a program that allows users to input a parcel tracking number. Store the tracking number and Status in 2d String Array. Initialize the array with values. Then, simulate the tracking process by displaying messages like "Parcel in transit," "Parcel out for delivery," or "Parcel delivered" based on the tracking number's status.

10. **Customer Data Validation:** Write a function which takes 2 parameters, data-denotes the data and detail-denotes if it is name address or phone number. Validate customer information based on following criteria. Ensure that names contain only letters and are properly capitalized, addresses do not contain special characters, and phone numbers follow a specific format (e.g., ###-###-####).

11. **Address Formatting:** Develop a function that takes an address as input (street, city, state, zip code) and formats it correctly, including capitalizing the first letter of each word and properly formatting the zip code.

12. **Order Confirmation Email:** Create a program that generates an order confirmation email. The email should include details such as the customer's name, order number, delivery address, and expected delivery date.

13. **Calculate Shipping Costs:** Develop a function that calculates the shipping cost based on the distance between two locations and the weight of the parcel. You can use string inputs for the source and destination addresses.

14. **Password Generator:** Create a function that generates secure passwords for courier system accounts. Ensure the passwords contain a mix of uppercase letters, lowercase letters, numbers, and special characters.

15. **Find Similar Addresses:** Implement a function that finds similar addresses in the system. This can be useful for identifying duplicate customer entries or optimizing delivery routes. Use string functions to implement this.



Following tasks are incremental stages to build an application and should be done in a single project

#### Task 5: Object Oriented Programming

##### Scope : Entity classes/Models/POJO, Abstraction/Encapsulation

Create the following **model/entity classes** within package **entities** with variables declared private, constructors(default and parametrized, getters, setters and toString())

##### 1. User Class:

**Variables:**

userID , userName , email , password , contactNumber , address

##### 2. Courier Class

**Variables:** courierID , senderName , senderAddress , receiverName , receiverAddress , weight , status, trackingNumber , deliveryDate , userID

##### 3. Employee Class:

**Variables** employeeID , employeeName , email , contactNumber , role String, salary

##### 4. Location Class

**Variables** LocationID , LocationName , Address

##### 5. CourierCompany Class

**Variables** companyName , courierDetails -collection of Courier Objects, employeeDetails- collection of Employee Objects, locationDetails - collection of Location Objects.

##### 6. Payment Class:

**Variables** PaymentID long, CourierID long, Amount double, PaymentDate Date

#### Task 6: Service Provider Interface /Abstract class

Create 2 **Interface /Abstract class** **ICourierUserService** and **ICourierAdminService** interface

**ICourierUserService {**

**// Customer-related functions**

**placeOrder()**

/\*\* Place a new courier order.

\* @param courierObj Courier object created using values entered by users

\* @return The unique tracking number for the courier order .

Use a static variable to generate unique tracking number. Initialize the static variable in Courier class with some random value. Increment the static variable each time in the constructor to generate next values.

**getOrderStatus();**

/\*\*Get the status of a courier order.

\* @param trackingNumber The tracking number of the courier order.

\* @return The status of the courier order (e.g., **yetToTransit, In Transit, Delivered**).

\*/

**cancelOrder()**

/\*\* Cancel a courier order.

\* @param trackingNumber The tracking number of the courier order to be canceled.

\* @return True if the order was successfully canceled, false otherwise.\*/



```
getAssignedOrder();  
    /** Get a list of orders assigned to a specific courier staff member  
    * @param courierStaffId The ID of the courier staff member.  
    * @return A list of courier orders assigned to the staff member.*/  
  
// Admin functions  
ICourierAdminService  
int addCourierStaff(Employee obj);  
    /** Add a new courier staff member to the system.  
    * @param name The name of the courier staff member.  
    * @param contactNumber The contact number of the courier staff member.  
    * @return The ID of the newly added courier staff member.  
    */
```

### Task 7: Exception Handling

(Scope: User Defined Exception/Checked /Unchecked Exception/Exception handling using try..catch finally,throw & throws keyword usage)

Define the following custom exceptions and throw them in methods whenever needed . Handle all the exceptions in main method,

1. **TrackingNumberNotFoundException** :throw this exception when user try to withdraw amount or transfer amount to another account
2. **InvalidEmployeeIdException** throw this exception when id entered for the employee not existing in the system

### Task 8: Collections

Scope: ArrayList/HashMap

Task: Improve the Courier Management System by using Java collections:

1. Create a new model named **CourierCompanyCollection** in **entity** package replacing the **Array of Objects** with List to accommodate dynamic updates in the **CourierCompany** class
2. Create a new implementation class **CourierUserServiceCollectionImpl** class in package **dao** which implements **ICourierUserService** interface which holds a variable named companyObj of type **CourierCompanyCollection**



### Task 8: Service implementation

1. Create **CourierUserServiceImpl** class which implements **ICourierUserService** interface which holds a variable named **companyObj** of type **CourierCompany**. This variable can be used to access the Object Arrays to access data relevant in method implementations.
2. Create **CourierAdminService Impl** class which inherits from **CourierUserServiceImpl** and implements **ICourierAdminService** interface.
3. Create **CourierAdminServiceCollectionImpl** class which inherits from **CourierUserServiceCollectionImpl** and implements **ICourierAdminService** interface.

### Task 9: Database Interaction

Connect your application to the SQL database for the Courier Management System

1. Write code to establish a connection to your SQL database.

Create a class **DBConnection** in a package **connectionutil** with a static variable **connection** of Type **Connection** and a static method **getConnection()** which returns connection.

Connection properties supplied in the connection string should be read from a property file.

2. Create a Service class **CourierServiceDb** in **dao** with a static variable named **connection** of type **Connection** which can be assigned in the constructor by invoking the method in **DBConnection** Class.

3. Include methods to **insert, update, and retrieve data** from the database (e.g., **inserting a new order, updating courier status**).

4. Implement a feature to retrieve and display the delivery history of a specific parcel by querying the database. 1. Generate and display reports using data retrieved from the database (e.g., **shipment status report, revenue report**).



Defining the relationship between the tables

TABLE NAME	RELATIONSHIP	RELATED TABLE	FOREIGN KEY
USER	ONE TO MANY	COURIER	UserID
COURIER	MANY TO ONE	USER	UserID
COURIER	MANY TO ONE	COURIERSERVICES	ServiceID
COURIER	MANY TO ONE	EMPLOYEE	EmployeeID
COURIER	ONE TO ONE	PAYMENT	CourierID
COURIERSERVICES	ONE TO MANY	COURIER	ServiceID
EMPLOYEE	ONE TO MANY	COURIER	EmployeeID
LOCATION	ONE TO MANY	PAYMENT	LocationID
PAYMENT	MANY TO ONE	COURIER	CourierID
PAYMENT	MANY TO ONE	LOCATION	LocationID

## Task - 1 Creating Database, Tables for the given entities and inserting sample data into the tables

Query 1

```
1 • CREATE DATABASE CouriersManagementSystem;
2   USE CouriersManagementSystem;
3
4 • CREATE TABLE User (
5     UserID INT PRIMARY KEY AUTO_INCREMENT,
6     Name VARCHAR(255) NOT NULL,
7     Email VARCHAR(255) UNIQUE NOT NULL,
8     Password VARCHAR(255) NOT NULL,
9     ContactNumber VARCHAR(20),
10    Address TEXT
11  );
12
13 • CREATE TABLE CourierServices (
14     ServiceID INT PRIMARY KEY AUTO_INCREMENT,
15     ServiceName VARCHAR(100) NOT NULL,
16     Cost DECIMAL(8, 2) NOT NULL
17  );
18
19
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 1	13:54:28	CREATE DATABASE CouriersManagementSystem	1 row(s) affected	0.015 sec
✓ 2	13:54:28	USE CouriersManagementSystem	0 row(s) affected	0.000 sec
✓ 3	13:56:08	CREATE TABLE User ( UserID INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(255) NOT NUL...	0 row(s) affected	0.031 sec
✓ 4	13:58:18	CREATE TABLE CourierServices ( ServiceID INT PRIMARY KEY AUTO_INCREMENT, ServiceName VAR...	0 row(s) affected	0.062 sec

Query 1

Limit to 1000 rows

```
16     Cost DECIMAL(8, 2) NOT NULL
17 );
18
19 CREATE TABLE Courier (
20     CourierID INT PRIMARY KEY AUTO_INCREMENT,
21     SenderName VARCHAR(255) NOT NULL,
22     SenderAddress TEXT NOT NULL,
23     ReceiverName VARCHAR(255) NOT NULL,
24     ReceiverAddress TEXT NOT NULL,
25     Weight DECIMAL(5, 2) NOT NULL,
26     Status VARCHAR(50) DEFAULT 'Pending',
27     TrackingNumber VARCHAR(20) UNIQUE NOT NULL,
28     DeliveryDate DATE,
29     UserID INT,
30     ServiceID INT,
31     FOREIGN KEY (UserID) REFERENCES User(UserID),
32     FOREIGN KEY (ServiceID) REFERENCES CourierServices(ServiceID)
33 );
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 1	13:54:28	CREATE DATABASE CouriersManagementSystem	1 row(s) affected	0.015 sec
✓ 2	13:54:28	USE CouriersManagementSystem	0 row(s) affected	0.000 sec
✓ 3	13:56:08	CREATE TABLE User ( UserID INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(255) NOT NUL...	0 row(s) affected	0.031 sec
✓ 4	13:58:18	CREATE TABLE CourierServices ( ServiceID INT PRIMARY KEY AUTO_INCREMENT, ServiceName VAR...	0 row(s) affected	0.062 sec
✓ 5	14:04:45	CREATE TABLE Courier ( CourierID INT PRIMARY KEY AUTO_INCREMENT, SenderName VARCHAR(25...	0 row(s) affected	0.093 sec

Query 1

Limit to 1000 rows

```
29     UserID INT,  
30     ServiceID INT,  
31     FOREIGN KEY (UserID) REFERENCES User(UserID),  
32     FOREIGN KEY (ServiceID) REFERENCES CourierServices(ServiceID)  
33 );  
34  
35 • CREATE TABLE Employee (  
36     EmployeeID INT PRIMARY KEY AUTO_INCREMENT,  
37     Name VARCHAR(255) NOT NULL,  
38     Email VARCHAR(255) UNIQUE NOT NULL,  
39     ContactNumber VARCHAR(20),  
40     Role VARCHAR(50) NOT NULL,  
41     Salary DECIMAL(10, 2) NOT NULL  
42 );  
43  
44 • CREATE TABLE Location (  
45     LocationID INT PRIMARY KEY AUTO_INCREMENT,  
46     LocationName VARCHAR(100) NOT NULL,  
47     Address TEXT NOT NULL  
48 );  
49
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 2	13:54:28	USE CouriersManagementSystem	0 row(s) affected	0.000 sec
✓ 3	13:56:08	CREATE TABLE User ( UserID INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(255) NOT N...	0 row(s) affected	0.031 sec
✓ 4	13:58:18	CREATE TABLE CourierServices ( ServiceID INT PRIMARY KEY AUTO_INCREMENT, ServiceName VA...	0 row(s) affected	0.062 sec
✓ 5	14:04:45	CREATE TABLE Courier ( CourierID INT PRIMARY KEY AUTO_INCREMENT, SenderName VARCHAR(2...	0 row(s) affected	0.093 sec
✓ 6	14:09:44	CREATE TABLE Employee ( EmployeeID INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(25...	0 row(s) affected	0.016 sec
✓ 7	14:10:48	CREATE TABLE Location ( LocationID INT PRIMARY KEY AUTO_INCREMENT, LocationName VARCH...	0 row(s) affected	0.031 sec

Query 1



Limit to 1000 rows



```
46     LocationName VARCHAR(100) NOT NULL,  
47     Address TEXT NOT NULL  
48 );  
49  
50 • CREATE TABLE Payment (  
51     PaymentID INT PRIMARY KEY AUTO_INCREMENT,  
52     CourierID INT,  
53     LocationID INT,  
54     Amount DECIMAL(10, 2) NOT NULL,  
55     PaymentDate DATE,  
56     FOREIGN KEY (CourierID) REFERENCES Courier(CourierID),  
57     FOREIGN KEY (LocationID) REFERENCES Location(LocationID)  
58 );  
59  
60  
61  
62  
63  
64  
65
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 3	13:56:08	CREATE TABLE User ( UserID INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(255) NOT N...	0 row(s) affected	0.031 sec
✓ 4	13:58:18	CREATE TABLE CourierServices ( ServiceID INT PRIMARY KEY AUTO_INCREMENT, ServiceName VA...	0 row(s) affected	0.062 sec
✓ 5	14:04:45	CREATE TABLE Courier ( CourierID INT PRIMARY KEY AUTO_INCREMENT, SenderName VARCHAR(2...	0 row(s) affected	0.093 sec
✓ 6	14:09:44	CREATE TABLE Employee ( EmployeeID INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(25...	0 row(s) affected	0.016 sec
✓ 7	14:10:48	CREATE TABLE Location ( LocationID INT PRIMARY KEY AUTO_INCREMENT, LocationName VARCH...	0 row(s) affected	0.031 sec
✓ 8	14:12:56	CREATE TABLE Payment ( PaymentID INT PRIMARY KEY AUTO_INCREMENT, CourierID INT, Locati...	0 row(s) affected	0.032 sec



Query 1

Limit to 1000 rows

```
56 FOREIGN KEY (CourierID) REFERENCES Courier(CourierID),
57 FOREIGN KEY (LocationID) REFERENCES Location(LocationID)
58 );
59
60 • INSERT INTO User (Name, Email, Password, ContactNumber, Address) VALUES
61 ('Chhota Bheem', 'bheem@dholakpur.com', 'bheem123', '9876543210', 'Dholakpur Village'),
62 ('Shinchan Nohara', 'shinchan@kasukabe.com', 'shinchan123', '9876543211', 'Kasukabe, Japan'),
63 ('Doraemon', 'doraemon@future.com', 'doraemon123', '9876543212', 'Tokyo, Japan'),
64 ('Ninja Hattori', 'hattori@ninja.com', 'hattori123', '9876543213', 'Shinobi Village'),
65 ('Jackie Chan', 'jackie@kungfu.com', 'jackie123', '9876543214', 'Hong Kong'),
66 ('Heidi', 'heidi@alps.com', 'heidi123', '9876543215', 'Swiss Alps'),
67 ('Raju', 'raju@dholakpur.com', 'raju123', '9876543216', 'Dholakpur Village'),
68 ('Chutki', 'chutki@dholakpur.com', 'chutki123', '9876543217', 'Dholakpur Village'),
69 ('Kalia', 'kalia@dholakpur.com', 'kalia123', '9876543218', 'Dholakpur Village'),
70 ('Indumati', 'indu@dholakpur.com', 'indu123', '9876543219', 'Dholakpur Village'),
71 ('Nobita Nobi', 'nobita@future.com', 'nobita123', '9876543220', 'Tokyo, Japan'),
72 ('Shizuka', 'shizuka@future.com', 'shizuka123', '9876543221', 'Tokyo, Japan'),
73 ('Gian', 'gian@future.com', 'gian123', '9876543222', 'Tokyo, Japan'),
74 ('Suneo', 'suneo@future.com', 'suneo123', '9876543223', 'Tokyo, Japan'),
75 ('Dekisugi', 'dehisugi@future.com', 'dehisugi123', '9876543224', 'Tokyo, Japan');
76
```

Output

Action Output


#	Time	Action	Message	Duration / Fetch
✓ 4	13:58:18	CREATE TABLE CourierServices ( ServiceID INT PRIMARY KEY AUTO_INCREMENT, ServiceName VA...	0 row(s) affected	0.062 sec
✓ 5	14:04:45	CREATE TABLE Courier ( CourierID INT PRIMARY KEY AUTO_INCREMENT, SenderName VARCHAR(2...	0 row(s) affected	0.093 sec
✓ 6	14:09:44	CREATE TABLE Employee ( EmployeeID INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(25...	0 row(s) affected	0.016 sec
✓ 7	14:10:48	CREATE TABLE Location ( LocationID INT PRIMARY KEY AUTO_INCREMENT, LocationName VARCH...	0 row(s) affected	0.031 sec
✓ 8	14:12:56	CREATE TABLE Payment ( PaymentID INT PRIMARY KEY AUTO_INCREMENT, CourierID INT, Locati...	0 row(s) affected	0.032 sec
✓ 9	14:20:46	INSERT INTO User (Name, Email, Password, ContactNumber, Address) VALUES ('Chhota Bheem', 'bheem@dh...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.015 sec

Query 1

 Limit to 1000 rows

```
73 ('Gian', 'gian@future.com', 'gian123', '9876543222', 'Tokyo, Japan'),
74 ('Suneo', 'suneo@future.com', 'suneo123', '9876543223', 'Tokyo, Japan'),
75 ('Dekisugi', 'dehisugi@future.com', 'dehisugi123', '9876543224', 'Tokyo, Japan');
76
77 • INSERT INTO CourierServices (ServiceName, Cost) VALUES
78 ('Standard Delivery', 100.00),
79 ('Express Delivery', 250.00),
80 ('Overnight Delivery', 500.00),
81 ('Same-Day Delivery', 700.00),
82 ('International Delivery', 1500.00),
83 ('Heavy Parcel Delivery', 1200.00),
84 ('Fragile Item Delivery', 800.00),
85 ('Medical Supply Delivery', 900.00),
86 ('Food Delivery', 300.00),
87 ('Document Delivery', 200.00),
88 ('Electronics Delivery', 1000.00),
89 ('Gift Delivery', 600.00),
90 ('Bulk Delivery', 2000.00),
91 ('Eco-Friendly Delivery', 400.00),
92 ('VIP Delivery', 3000.00);
93
```

Output

 Action Output

#	Time	Action	Message	Duration / Fetch
✓ 5	14:04:45	CREATE TABLE Courier ( CourierID INT PRIMARY KEY AUTO_INCREMENT, SenderName VARCHAR(255), LocationID INT, Cost INT);	0 row(s) affected	0.093 sec
✓ 6	14:09:44	CREATE TABLE Employee ( EmployeeID INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(255), LocationID INT, Salary INT);	0 row(s) affected	0.016 sec
✓ 7	14:10:48	CREATE TABLE Location ( LocationID INT PRIMARY KEY AUTO_INCREMENT, LocationName VARCHAR(255), Address VARCHAR(255));	0 row(s) affected	0.031 sec
✓ 8	14:12:56	CREATE TABLE Payment ( PaymentID INT PRIMARY KEY AUTO_INCREMENT, CourierID INT, LocationID INT, Amount INT);	0 row(s) affected	0.032 sec
✓ 9	14:20:46	INSERT INTO User (Name, Email, Password, ContactNumber, Address) VALUES ('Chhota Bheem', 'bheem@dhoni.com', 'bheem123', '9876543210', 'Mumbai, India');	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.015 sec
✓ 10	14:21:49	INSERT INTO CourierServices (ServiceName, Cost) VALUES ('Standard Delivery', 100.00), ('Express Delivery', 250.00), ('Overnight Delivery', 500.00), ('Same-Day Delivery', 700.00), ('International Delivery', 1500.00), ('Heavy Parcel Delivery', 1200.00), ('Fragile Item Delivery', 800.00), ('Medical Supply Delivery', 900.00), ('Food Delivery', 300.00), ('Document Delivery', 200.00), ('Electronics Delivery', 1000.00), ('Gift Delivery', 600.00), ('Bulk Delivery', 2000.00), ('Eco-Friendly Delivery', 400.00), ('VIP Delivery', 3000.00);	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.000 sec



Query 1



Limit to 1000 rows

```
92 ('VIP Delivery', 3000.00);
93
94 • INSERT INTO Courier (SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate, UserID, ServiceID) VALUES
95 ('Chhota Bheem', 'Dholakpur Village', 'Shinchan Nohara', 'Kasukabe, Japan', 2.5, 'In Transit', 'TRK123456', '2023-10-25', 1, 1),
96 ('Doraemon', 'Tokyo, Japan', 'Ninja Hattori', 'Shinobi Village', 1.8, 'Delivered', 'TRK123457', '2023-10-20', 3, 2),
97 ('Heidi', 'Swiss Alps', 'Jackie Chan', 'Hong Kong', 3.0, 'Pending', 'TRK123458', '2023-10-30', 6, 3),
98 ('Raju', 'Dholakpur Village', 'Chutki', 'Dholakpur Village', 5.0, 'In Transit', 'TRK123459', '2023-10-26', 7, 4),
99 ('Kalia', 'Dholakpur Village', 'Indumati', 'Dholakpur Village', 4.2, 'Delivered', 'TRK123460', '2023-10-22', 9, 5),
100 ('Nobita Nobi', 'Tokyo, Japan', 'Shizuka Minamoto', 'Tokyo, Japan', 1.5, 'Pending', 'TRK123461', '2023-10-31', 11, 6),
101 ('Gian', 'Tokyo, Japan', 'Suneo Honekawa', 'Tokyo, Japan', 6.0, 'In Transit', 'TRK123462', '2023-10-27', 13, 7),
102 ('Dekisugi', 'Tokyo, Japan', 'Nobita Nobi', 'Tokyo, Japan', 2.0, 'Delivered', 'TRK123463', '2023-10-23', 15, 8),
103 ('Shinchan Nohara', 'Kasukabe, Japan', 'Doraemon', 'Tokyo, Japan', 3.5, 'Pending', 'TRK123464', '2023-11-01', 2, 9),
104 ('Jackie Chan', 'Hong Kong', 'Heidi', 'Swiss Alps', 7.0, 'In Transit', 'TRK123465', '2023-10-28', 5, 10),
105 ('Chutki', 'Dholakpur Village', 'Raju', 'Dholakpur Village', 1.0, 'Delivered', 'TRK123466', '2023-10-24', 8, 11),
106 ('Indumati', 'Dholakpur Village', 'Kalia', 'Dholakpur Village', 2.3, 'Pending', 'TRK123467', '2023-11-02', 10, 12),
107 ('Shizuka', 'Tokyo, Japan', 'Gian', 'Tokyo, Japan', 4.7, 'In Transit', 'TRK123468', '2023-10-29', 12, 13),
108 ('Suneo', 'Tokyo, Japan', 'Dekisugi', 'Tokyo, Japan', 3.8, 'Delivered', 'TRK123469', '2023-10-21', 14, 14),
109 ('Ninja Hattori', 'Shinobi Village', 'Chhota Bheem', 'Dholakpur Village', 5.5, 'Pending', 'TRK123470', '2023-11-03', 4, 15);
110
111
112
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 6	14:09:44	CREATE TABLE Employee ( EmployeeID INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(25...	0 row(s) affected	0.016 sec
✓ 7	14:10:48	CREATE TABLE Location ( LocationID INT PRIMARY KEY AUTO_INCREMENT, LocationName VARCH...	0 row(s) affected	0.031 sec
✓ 8	14:12:56	CREATE TABLE Payment ( PaymentID INT PRIMARY KEY AUTO_INCREMENT, CourierID INT, Locati...	0 row(s) affected	0.032 sec
✓ 9	14:20:46	INSERT INTO User (Name, Email, Password, ContactNumber, Address) VALUES ('Chhota Bheem', 'bheem@dh...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.015 sec
✓ 10	14:21:49	INSERT INTO CourierServices (ServiceName, Cost) VALUES ('Standard Delivery', 100.00), ('Express Delivery', ...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.000 sec
✓ 11	14:30:20	INSERT INTO Courier (SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, Tracki...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.015 sec



Query 1

 Limit to 1000 rows

```
109 ('Ninja Hattori', 'Shinobi Village', 'Chhota Bheem', 'Dholakpur Village', 5.5, 'Pending', 'TRK123470', '2023-11-03', 4, 15);
```

110

```
111 • INSERT INTO Employee (Name, Email, ContactNumber, Role, Salary) VALUES
```

```
112 ('Raju', 'raju@dholakpur.com', '9876543220', 'Delivery Boy', 20000.00),
```

```
113 ('Jaggu', 'jaggu@dholakpur.com', '9876543221', 'Manager', 50000.00),
```

```
114 ('Kalia', 'kalia@dholakpur.com', '9876543222', 'Warehouse Staff', 15000.00),
```

```
115 ('Indumati', 'indu@dholakpur.com', '9876543223', 'Customer Support', 18000.00),
```

```
116 ('Chutki', 'chutki@dholakpur.com', '9876543224', 'Delivery Boy', 20000.00),
```

```
117 ('Nobita Nobi', 'nobita@future.com', '9876543225', 'IT Support', 25000.00),
```

```
118 ('Shizuka', 'shizuka@future.com', '9876543226', 'HR Manager', 40000.00),
```

```
119 ('Gian', 'gian@future.com', '9876543227', 'Security', 22000.00),
```

```
120 ('Suneo', 'suneo@future.com', '9876543228', 'Accountant', 30000.00),
```

```
121 ('Dekisugi', 'dehisugi@future.com', '9876543229', 'Developer', 35000.00),
```

```
122 ('Heidi', 'heidi@alps.com', '9876543230', 'Delivery Boy', 20000.00),
```

```
123 ('Jackie Chan', 'jackie@kungfu.com', '9876543231', 'Security', 22000.00),
```

```
124 ('Ninja Hattori', 'hattori@ninja.com', '9876543232', 'Delivery Boy', 20000.00),
```

```
125 ('Doraemon', 'doraemon@future.com', '9876543233', 'IT Support', 25000.00),
```

```
126 ('Shinchan Nohara', 'shinchan@kasukabe.com', '9876543234', 'Customer Support', 18000.00);
```

127

128

129

Output

 Action Output

#	Time	Action	Message	Duration / Fetch
✓ 1	17:05:23	USE CouriersManagementSystem	0 row(s) affected	0.000 sec
✓ 2	17:05:35	INSERT INTO Employee (Name, Email, ContactNumber, Role, Salary) VALUES ('Raju', 'raju@dholakpur.com', '98...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.063 sec

Query 1

Limit to 1000 rows

```
126 ('Shinchan Nohara', 'shinchan@kasukabe.com', '9876543234', 'Customer Support', 18000.00);
127
128 • INSERT INTO Location (LocationName, Address) VALUES
129 ('Dholakpur Branch', 'Dholakpur Village, India'),
130 ('Kasukabe Branch', 'Kasukabe, Japan'),
131 ('Tokyo Branch', 'Tokyo, Japan'),
132 ('Hong Kong Branch', 'Hong Kong'),
133 ('Swiss Alps Branch', 'Swiss Alps'),
134 ('Mumbai Branch', 'Mumbai, India'),
135 ('Delhi Branch', 'Delhi, India'),
136 ('Bangalore Branch', 'Bangalore, India'),
137 ('Chennai Branch', 'Chennai, India'),
138 ('Kolkata Branch', 'Kolkata, India'),
139 ('New York Branch', 'New York, USA'),
140 ('London Branch', 'London, UK'),
141 ('Paris Branch', 'Paris, France'),
142 ('Sydney Branch', 'Sydney, Australia'),
143 ('Dubai Branch', 'Dubai, UAE');
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 1	17:05:23	USE CouriersManagementSystem	0 row(s) affected	0.000 sec
✓ 2	17:05:35	INSERT INTO Employee (Name, Email, ContactNumber, Role, Salary) VALUES ('Raju', 'raju@dholakpur.com', '98...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.063 sec
✓ 3	17:09:56	INSERT INTO Location (LocationName, Address) VALUES ('Dholakpur Branch', 'Dholakpur Village, India'), ('Kasu...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.062 sec

Query 1



```
141 ('Paris Branch', 'Paris, France'),
142 ('Sydney Branch', 'Sydney, Australia'),
143 ('Dubai Branch', 'Dubai, UAE');
144
145 • INSERT INTO Payment (CourierID, LocationID, Amount, PaymentDate) VALUES
146 (1, 1, 100.00, '2023-10-24'),
147 (2, 2, 250.00, '2023-10-19'),
148 (3, 3, 500.00, '2023-10-29'),
149 (4, 4, 700.00, '2023-10-26'),
150 (5, 5, 1500.00, '2023-10-22'),
151 (6, 6, 1200.00, '2023-10-31'),
152 (7, 7, 800.00, '2023-10-27'),
153 (8, 8, 900.00, '2023-10-23'),
154 (9, 9, 300.00, '2023-11-01'),
155 (10, 10, 200.00, '2023-10-28'),
156 (11, 11, 1000.00, '2023-10-24'),
157 (12, 12, 600.00, '2023-11-02'),
158 (13, 13, 2000.00, '2023-10-29'),
159 (14, 14, 400.00, '2023-10-21'),
160 (15, 15, 3000.00, '2023-11-03');
161
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 1	17:05:23	USE CouriersManagementSystem	0 row(s) affected	0.000 sec
✓ 2	17:05:35	INSERT INTO Employee (Name, Email, ContactNumber, Role, Salary) VALUES ('Raju', 'raju@dhola...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.063 sec
✓ 3	17:09:56	INSERT INTO Location (LocationName, Address) VALUES ('Dholakpur Branch', 'Dholakpur Village, India'), ('Kasu...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.062 sec
✓ 4	17:12:06	INSERT INTO Payment (CourierID, LocationID, Amount, PaymentDate) VALUES (1, 1, 100.00, '2023-10-24'), (2, ...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.016 sec

## Task - 2 Select,Where

### 1. List all customers:

Query 1 x

Limit to 1000 rows

```
157 (12, 12, 600.00, '2023-11-02'),
158 (13, 13, 2000.00, '2023-10-29'),
159 (14, 14, 400.00, '2023-10-21'),
160 (15, 15, 3000.00, '2023-11-03');
161
162 • SELECT * FROM User;
163
```

Result Grid

UserID	Name	Email	Password	ContactNumber	Address
1	Chhota Bheem	bheem@dholakpur.com	bheem123	9876543210	Dholakpur Village
2	Shinchan Nohara	shinchan@kasukabe.com	shinchan123	9876543211	Kasukabe, Japan
3	Doraemon	doraemon@future.com	doraemon123	9876543212	Tokyo, Japan
4	Ninja Hattori	hattori@ninja.com	hattori123	9876543213	Shinobi Village
5	Jackie Chan	jackie@kungfu.com	jackie123	9876543214	Hong Kong
6	Heidi	heidi@alps.com	heidi123	9876543215	Swiss Alps
7	Raju	raju@dholakpur.com	raju123	9876543216	Dholakpur Village
8	Chutki	chutki@dholakpur.com	chutki123	9876543217	Dholakpur Village
9	Kalia	kalia@dholakpur.com	kalia123	9876543218	Dholakpur Village
10	Indumati	indu@dholakpur.com	indu123	9876543219	Dholakpur Village
11	Nobita Nobi	nobita@future.com	nobita123	9876543220	Tokyo, Japan
12	Shizuka	shizuka@future.com	shizuka123	9876543221	Tokyo, Japan
13	Gian	gian@future.com	gian123	9876543222	Tokyo, Japan

User 1 x

Apply Revert

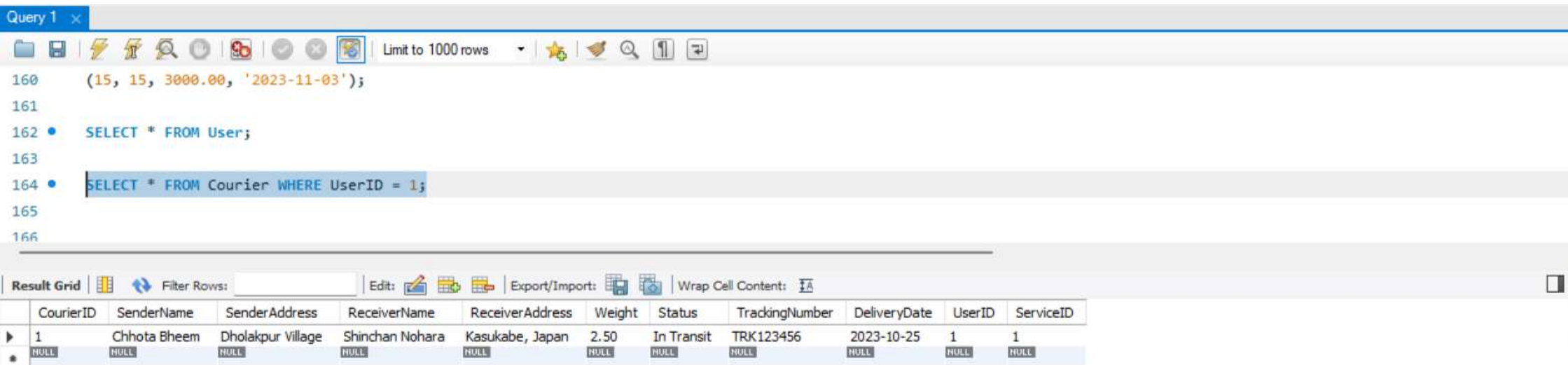
Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 1	17:05:23	USE CouriersManagementSystem	0 row(s) affected	0.000 sec
✓ 2	17:05:35	INSERT INTO Employee (Name, Email, ContactNumber, Role, Salary) VALUES ('Raju', 'raju@dholakpur.com', '9...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.063 sec
✓ 3	17:09:56	INSERT INTO Location (LocationName, Address) VALUES ('Dholakpur Branch', 'Dholakpur Village, India'), ('Kas...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.062 sec
✓ 4	17:12:06	INSERT INTO Payment (CourierID, LocationID, Amount, PaymentDate) VALUES (1, 1, 100.00, '2023-10-24'), (2,...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.016 sec
✓ 5	17:55:21	SELECT * FROM User LIMIT 0, 1000	15 row(s) returned	0.000 sec / 0.000 sec



## 2. List all orders for a specific customer:

[illegible]

### 3. List all couriers:

Query 1 x

Limit to 1000 rows

```
161
162 • SELECT * FROM User;
163
164 • SELECT * FROM Courier WHERE UserID = 1;
165
166 • SELECT * FROM Courier;
167
168
```

Result Grid		Filter Rows:		Edit:		Export/Import:		Wrap Cell Content: <div></div>			
	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	UserID	ServiceID
	1	Chhota Bheem	Dholakpur Village	Shinchan Nohara	Kasukabe, Japan	2.50	In Transit	TRK123456	2023-10-25	1	1
	2	Doraemon	Tokyo, Japan	Ninja Hattori	Shinobi Village	1.80	Delivered	TRK123457	2023-10-20	3	2
	3	Heidi	Swiss Alps	Jackie Chan	Hong Kong	3.00	Pending	TRK123458	2023-10-30	6	3
	4	Raju	Dholakpur Village	Chutki	Dholakpur Village	5.00	In Transit	TRK123459	2023-10-26	7	4
	5	Kalia	Dholakpur Village	Indumati	Dholakpur Village	4.20	Delivered	TRK123460	2023-10-22	9	5
	6	Nobita Nobi	Tokyo, Japan	Shizuka Minamoto	Tokyo, Japan	1.50	Pending	TRK123461	2023-10-31	11	6
	7	Gian	Tokyo, Japan	Suneo Honekawa	Tokyo, Japan	6.00	In Transit	TRK123462	2023-10-27	13	7
	8	Dekisugi	Tokyo, Japan	Nobita Nobi	Tokyo, Japan	2.00	Delivered	TRK123463	2023-10-23	15	8
	9	Shinchan Nohara	Kasukabe, Japan	Doraemon	Tokyo, Japan	3.50	Pending	TRK123464	2023-11-01	2	9
	10	Jackie Chan	Hong Kong	Heidi	Swiss Alps	7.00	In Transit	TRK123465	2023-10-28	5	10
	11	Chutki	Dholakpur Village	Raju	Dholakpur Village	1.00	Delivered	TRK123466	2023-10-24	8	11
	12	Indumati	Dholakpur Village	Kalia	Dholakpur Village	2.30	Pending	TRK123467	2023-11-02	10	12
	13	Shizuka	Tokyo, Japan	Gian	Tokyo, Japan	4.70	In Transit	TRK123468	2023-10-29	12	13
	14	Suneo	Tokyo, Japan	Dekisugi	Tokyo, Japan	3.80	Delivered	TRK123469	2023-10-21	14	14
	15	Ninja Hattori	Shinobi Village	Chhota Bheem	Dholakpur Village	5.50	Pending	TRK123470	2023-11-03	4	15

#### 4. List all packages for a specific order:

[illegible]

```
L70 • SELECT * FROM Courier WHERE CourierID = 1;
```

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	UserID	ServiceID
1	Chhotabheem	Dholakpur Village	Shinchannohara	Kasukabe, Japan	2.50	In Transit	TRK123456	2023-10-25	1	1



```
171
172 • SELECT * FROM Courier WHERE Status != 'Delivered';
173
```

[illegible]

7. List all packages that are scheduled for delivery today:

173

```
174 • SELECT * FROM Courier WHERE DeliveryDate = '2023-10-25';
```

474




[illegible]

## 457

[illegible]

9. Calculate the total number of packages for each courier.



```
177  
178 • SELECT CourierID, COUNT(*) AS TotalPackages  
179 FROM Courier  
180 GROUP BY CourierID;
```

Result Grid   Filter Rows:  Export:  Wrap Cell Content: 

	CourierID	TotalPackages
▶	1	1
	2	1
	3	1
	4	1
	5	1
	6	1
	7	1
	8	1
	9	1
	10	1
	11	1
	12	1
	13	1
	14	1
	15	1

## 10. Find the average delivery time for each courier

```
196 • SELECT CourierID, AVG(DATEDIFF(DeliveryDate, OrderDate)) AS AvgDeliveryTime
197 FROM Courier
198 GROUP BY CourierID;
```

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

CourierID	AvgDeliveryTime
1	2.0000
2	2.0000
3	2.0000
4	2.0000
5	2.0000
6	2.0000
7	2.0000
8	2.0000
9	2.0000
10	2.0000
11	2.0000
12	2.0000
13	2.0000
14	2.0000
15	2.0000

203  
204  
205  
206  
207

```
SELECT *  
FROM Courier  
WHERE Weight BETWEEN 2.0 AND 5.0;
```

[illegible]



## 12. Retrieve employees whose names contain 'John'

```
215  
216 • SELECT *  
217 FROM Employee  
218 WHERE Name LIKE '%Raj%';  
219
```

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

EmployeeID	Name	Email	ContactNumber	Role	Salary
1	Raju	raju@dholakpur.com	9876543220	Delivery Boy	20000.00
NULL	NULL	NULL	NULL	NULL	NULL

### 13. Retrieve all courier records with payments greater than \$50.

```
220 SELECT *
221 FROM Courier
222 WHERE CourierID IN (
223     SELECT CourierID
224     FROM Payment
225     WHERE Amount > 50
226 )
```

Result Grid												
Filter Rows: <input type="text"/> Edit:    Export/Import:   Wrap Cell Content:												
	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	UserID	ServiceID	OrderDate
1	1	Chhota Bheem	Dholakpur Village	Shinchan Nohara	Kasukabe, Japan	2.50	In Transit	TRK123456	2023-10-25	1	1	2023-10-23
2	2	Doraemon	Tokyo, Japan	Ninja Hattori	Shinobi Village	1.80	Delivered	TRK123457	2023-10-20	3	2	2023-10-18
3	3	Heidi	Swiss Alps	Jackie Chan	Hong Kong	3.00	Pending	TRK123458	2023-10-30	6	3	2023-10-28
4	4	Raju	Dholakpur Village	Chutki	Dholakpur Village	5.00	In Transit	TRK123459	2023-10-26	7	4	2023-10-24
5	5	Kalia	Dholakpur Village	Indumati	Dholakpur Village	4.20	Delivered	TRK123460	2023-10-22	9	5	2023-10-20
6	6	Nobita Nobi	Tokyo, Japan	Shizuka Minamoto	Tokyo, Japan	1.50	Pending	TRK123461	2023-10-31	11	6	2023-10-29
7	7	Gian	Tokyo, Japan	Suneo Honekawa	Tokyo, Japan	6.00	In Transit	TRK123462	2023-10-27	13	7	2023-10-25
8	8	Dekisugi	Tokyo, Japan	Nobita Nobi	Tokyo, Japan	2.00	Delivered	TRK123463	2023-10-23	15	8	2023-10-21
9	9	Shinchan Nohara	Kasukabe, Japan	Doraemon	Tokyo, Japan	3.50	Pending	TRK123464	2023-11-01	2	9	2023-10-30
10	10	Jackie Chan	Hong Kong	Heidi	Swiss Alps	7.00	In Transit	TRK123465	2023-10-28	5	10	2023-10-26
11	11	Chutki	Dholakpur Village	Raju	Dholakpur Village	1.00	Delivered	TRK123466	2023-10-24	8	11	2023-10-22
12	12	Indumati	Dholakpur Village	Kalia	Dholakpur Village	2.30	Pending	TRK123467	2023-11-02	10	12	2023-10-31
13	13	Shizuka	Tokyo, Japan	Gian	Tokyo, Japan	4.70	In Transit	TRK123468	2023-10-29	12	13	2023-10-27
14	14	Suneo	Tokyo, Japan	Dekisugi	Tokyo, Japan	3.80	Delivered	TRK123469	2023-10-21	14	14	2023-10-19
15	15	Ninja Hattori	Shinobi Village	Chhota Bheem	Dholakpur Village	5.50	Pending	TRK123470	2023-11-03	4	15	2023-11-01

Courier 22 x



Task 3: GroupBy, Aggregate Functions, Having, Order By, where  
14. Find the total number of couriers handled by each employee.

240

241 • `SELECT EmployeeID, COUNT(*) AS TotalCouriersHandled`

242 `FROM Courier`

243 `GROUP BY EmployeeID;`

244

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

	EmployeeID	TotalCouriersHandled
•	1	1
	2	1
	3	1
	NULL	12

244 15. Calculate the total revenue generated by each location

245 • `SELECT LocationID, SUM(Amount) AS TotalRevenue`

246 `FROM Payment`

247 `GROUP BY LocationID;`

Result Grid



Filter Rows:

Export:



Wrap Cell Content:



	LocationID	TotalRevenue
1	1	100.00
2	2	250.00
3	3	500.00
4	4	700.00
5	5	1500.00
6	6	1200.00
7	7	800.00
8	8	900.00
9	9	300.00
10	10	200.00
11	11	1000.00
12	12	600.00
13	13	2000.00
14	14	400.00
15	15	3000.00

248 16. Find the total number of couriers delivered to each location.

```
249 • SELECT LocationID, COUNT(*) AS TotalCouriersDelivered
250 FROM Payment
251 GROUP BY LocationID;
```

Result Grid



Filter Rows:

Export:



Wrap Cell Content:



	LocationID	TotalCouriersDelivered
▶	1	1
	2	1
	3	1
	4	1
	5	1
	6	1
	7	1
	8	1
	9	1
	10	1
	11	1
	12	1
	13	1
	14	1
	15	1

17. Find the courier with the highest average delivery time:

```
252  
253 • SELECT CourierID, AVG(DATEDIFF(DeliveryDate, OrderDate)) AS AvgDeliveryTime  
254 FROM Courier  
255 GROUP BY CourierID  
256 ORDER BY AvgDeliveryTime DESC  
257 LIMIT 1;
```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
	CourierID	AvgDeliveryTime				
▶	1	2.0000				

## 18. Find Locations with Total Payments Less Than a Certain Amount

258

259 • `SELECT LocationID, SUM(Amount) AS TotalPayments`

260 `FROM Payment`

261 `GROUP BY LocationID`

262 `HAVING TotalPayments < 1000;`

263

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

	LocationID	TotalPayments
▶	1	100.00
	2	250.00
	3	500.00
	4	700.00
	7	800.00
	8	900.00
	9	300.00
	10	200.00
	12	600.00
	14	400.00

## 263 19. Calculate Total Payments per Location

```
264 • SELECT LocationID, SUM(Amount) AS TotalPayments  
265 FROM Payment  
266 GROUP BY LocationID;
```

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

	LocationID	TotalPayments
1	1	100.00
2	2	250.00
3	3	500.00
4	4	700.00
5	5	1500.00
6	6	1200.00
7	7	800.00
8	8	900.00
9	9	300.00
10	10	200.00
11	11	1000.00
12	12	600.00
13	13	2000.00
14	14	400.00
15	15	3000.00

20. Retrieve couriers who have received payments totaling more than \$1000 in a specific location (LocationID = X):

```
267  
268 • SELECT CourierID  
269 FROM Payment  
270 WHERE LocationID = 3  
271 GROUP BY CourierID  
272 HAVING SUM(Amount) > 1000;  
273
```





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

CourierID
-----------

21. Retrieve couriers who have received payments totaling more than \$1000 after a certain date (PaymentDate > 'YYYY-MM-DD'):

273  
274  
275  
276  
277  
278  
279

```
SELECT CourierID  
FROM Payment  
WHERE PaymentDate > '2023-10-25'  
GROUP BY CourierID  
HAVING SUM(Amount) > 1000;
```


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

	CourierID
▶	6
	13
	15



22. Retrieve locations where the total amount received is more than \$5000 before a certain date  
(PaymentDate > 'YYYY-MM-DD')

```
279
280 • SELECT LocationID, SUM(Amount) AS TotalAmount
281 FROM Payment
282 WHERE PaymentDate < '2023-10-30'
283 GROUP BY LocationID
284 HAVING TotalAmount > 5000;
```

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

LocationID	TotalAmount
------------	-------------

## Task 4: Inner Join, Full Outer Join, Cross Join, Left Outer Join, Right Outer Join

### 23. Retrieve Payments with Courier Information


285

286 • `SELECT Payment.*, Courier.SenderName, Courier.ReceiverName`

287 `FROM Payment`

288 `INNER JOIN Courier ON Payment.CourierID = Courier.CourierID;`




289

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

	PaymentID	CourierID	LocationID	Amount	PaymentDate	SenderName	ReceiverName
▶	1	1	1	100.00	2023-10-24	Chhota Bheem	Shinchan Nohara
	2	2	2	250.00	2023-10-19	Doraemon	Ninja Hattori
	3	3	3	500.00	2023-10-29	Heidi	Jackie Chan
	4	4	4	700.00	2023-10-26	Raju	Chutki
	5	5	5	1500.00	2023-10-22	Kalia	Indumati
	6	6	6	1200.00	2023-10-31	Nobita Nobi	Shizuka Minamoto
	7	7	7	800.00	2023-10-27	Gian	Suneo Honekawa
	8	8	8	900.00	2023-10-23	Dekisugi	Nobita Nobi
	9	9	9	300.00	2023-11-01	Shinchan Nohara	Doraemon
	10	10	10	200.00	2023-10-28	Jackie Chan	Heidi
	11	11	11	1000.00	2023-10-24	Chutki	Raju
	12	12	12	600.00	2023-11-02	Indumati	Kalia
	13	13	13	2000.00	2023-10-29	Shizuka	Gian
	14	14	14	400.00	2023-10-21	Suneo	Dekisugi
	15	15	15	3000.00	2023-11-03	Ninja Hattori	Chhota Bheem

## 24. Retrieve Payments with Location Information



```
289
290 • SELECT Payment.*, Location.LocationName
291 FROM Payment
292 INNER JOIN Location ON Payment.LocationID = Location.LocationID;
```

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

	PaymentID	CourierID	LocationID	Amount	PaymentDate	LocationName
▶	1	1	1	100.00	2023-10-24	Dholakpur Branch
	2	2	2	250.00	2023-10-19	Kasukabe Branch
	3	3	3	500.00	2023-10-29	Tokyo Branch
	4	4	4	700.00	2023-10-26	Hong Kong Branch
	5	5	5	1500.00	2023-10-22	Swiss Alps Branch
	6	6	6	1200.00	2023-10-31	Mumbai Branch
	7	7	7	800.00	2023-10-27	Delhi Branch
	8	8	8	900.00	2023-10-23	Bangalore Branch
	9	9	9	300.00	2023-11-01	Chennai Branch
	10	10	10	200.00	2023-10-28	Kolkata Branch
	11	11	11	1000.00	2023-10-24	New York Branch
	12	12	12	600.00	2023-11-02	London Branch
	13	13	13	2000.00	2023-10-29	Paris Branch
	14	14	14	400.00	2023-10-21	Sydney Branch
	15	15	15	3000.00	2023-11-03	Dubai Branch

## 25. Retrieve Payments with Courier and Location Information

```
293
294 • SELECT Payment.*, Courier.SenderName, Courier.ReceiverName, Location.LocationName
295 FROM Payment
296 INNER JOIN Courier ON Payment.CourierID = Courier.CourierID
297 INNER JOIN Location ON Payment.LocationID = Location.LocationID;
```

Result Grid   Filter Rows:  Export:  Wrap Cell Content: 

	PaymentID	CourierID	LocationID	Amount	PaymentDate	SenderName	ReceiverName	LocationName
1	1	1	1	100.00	2023-10-24	Chhota Bheem	Shinchan Nohara	Dholakpur Branch
2	2	2	2	250.00	2023-10-19	Doraemon	Ninja Hattori	Kasukabe Branch
3	3	3	3	500.00	2023-10-29	Heidi	Jackie Chan	Tokyo Branch
4	4	4	4	700.00	2023-10-26	Raju	Chutki	Hong Kong Branch
5	5	5	5	1500.00	2023-10-22	Kalia	Indumati	Swiss Alps Branch
6	6	6	6	1200.00	2023-10-31	Nobita Nobi	Shizuka Minamoto	Mumbai Branch
7	7	7	7	800.00	2023-10-27	Gian	Suneo Honekawa	Delhi Branch
8	8	8	8	900.00	2023-10-23	Dekisugi	Nobita Nobi	Bangalore Branch
9	9	9	9	300.00	2023-11-01	Shinchan Nohara	Doraemon	Chennai Branch
10	10	10	10	200.00	2023-10-28	Jackie Chan	Heidi	Kolkata Branch
11	11	11	11	1000.00	2023-10-24	Chutki	Raju	New York Branch
12	12	12	12	600.00	2023-11-02	Indumati	Kalia	London Branch
13	13	13	13	2000.00	2023-10-29	Shizuka	Gian	Paris Branch
14	14	14	14	400.00	2023-10-21	Suneo	Dekisugi	Sydney Branch
15	15	15	15	3000.00	2023-11-03	Ninja Hattori	Chhota Bheem	Dubai Branch




## 26. List all payments with courier details

```
299 • SELECT Payment.*, Courier.*
300 FROM Payment
301 INNER JOIN Courier ON Payment.CourierID = Courier.CourierID;
302
```

Result Grid																				
Filter Rows: <input type="text"/>																		Export:	Wrap Cell Content:	
	PaymentID	CourierID	LocationID	Amount	PaymentDate	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	UserID	ServiceID	Order			
▶	1	1	1	100.00	2023-10-24	1	Chhota Bheem	Dholakpur Village	Shinchan Nohara	Kasukabe, Japan	2.50	In Transit	TRK123456	2023-10-25	1	1	2023-			
	2	2	2	250.00	2023-10-19	2	Doraemon	Tokyo, Japan	Ninja Hattori	Shinobi Village	1.80	Delivered	TRK123457	2023-10-20	3	2	2023-			
	3	3	3	500.00	2023-10-29	3	Heidi	Swiss Alps	Jackie Chan	Hong Kong	3.00	Pending	TRK123458	2023-10-30	6	3	2023-			
	4	4	4	700.00	2023-10-26	4	Raju	Dholakpur Village	Chutki	Dholakpur Village	5.00	In Transit	TRK123459	2023-10-26	7	4	2023-			
	5	5	5	1500.00	2023-10-22	5	Kalia	Dholakpur Village	Indumati	Dholakpur Village	4.20	Delivered	TRK123460	2023-10-22	9	5	2023-			
	6	6	6	1200.00	2023-10-31	6	Nobita Nobi	Tokyo, Japan	Shizuka Minamoto	Tokyo, Japan	1.50	Pending	TRK123461	2023-10-31	11	6	2023-			
	7	7	7	800.00	2023-10-27	7	Gian	Tokyo, Japan	Suneo Honekawa	Tokyo, Japan	6.00	In Transit	TRK123462	2023-10-27	13	7	2023-			
	8	8	8	900.00	2023-10-23	8	Dekisugi	Tokyo, Japan	Nobita Nobi	Tokyo, Japan	2.00	Delivered	TRK123463	2023-10-23	15	8	2023-			
	9	9	9	300.00	2023-11-01	9	Shinchan Nohara	Kasukabe, Japan	Doraemon	Tokyo, Japan	3.50	Pending	TRK123464	2023-11-01	2	9	2023-			
	10	10	10	200.00	2023-10-28	10	Jackie Chan	Hong Kong	Heidi	Swiss Alps	7.00	In Transit	TRK123465	2023-10-28	5	10	2023-			
	11	11	11	1000.00	2023-10-24	11	Chutki	Dholakpur Village	Raju	Dholakpur Village	1.00	Delivered	TRK123466	2023-10-24	8	11	2023-			
	12	12	12	600.00	2023-11-02	12	Indumati	Dholakpur Village	Kalia	Dholakpur Village	2.30	Pending	TRK123467	2023-11-02	10	12	2023-			
	13	13	13	2000.00	2023-10-29	13	12 Juka	Tokyo, Japan	Gian	Tokyo, Japan	4.70	In Transit	TRK123468	2023-10-29	12	13	2023-			
	14	14	14	400.00	2023-10-21	14	Suneo	Tokyo, Japan	Dekisugi	Tokyo, Japan	3.80	Delivered	TRK123469	2023-10-21	14	14	2023-			
	15	15	15	3000.00	2023-11-03	15	Ninja Hattori	Shinobi Village	Chhota Bheem	Dholakpur Village	5.50	Pending	TRK123470	2023-11-03	4	15	2023-			

## 27. Total payments received for each courier

```
303 • SELECT Courier.CourierID, Courier.SenderName, SUM(Payment.Amount) AS TotalPayments
304 FROM Payment
305 INNER JOIN Courier ON Payment.CourierID = Courier.CourierID
306 GROUP BY Courier.CourierID;
307
```

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

	CourierID	SenderName	TotalPayments
▶	1	Chhota Bheem	100.00
	2	Doraemon	250.00
	3	Heidi	500.00
	4	Raju	700.00
	5	Kalia	1500.00
	6	Nobita Nobi	1200.00
	7	Gian	800.00
	8	Dekisugi	900.00
	9	Shinchan Nohara	300.00
	10	Jackie Chan	200.00
	11	Chutki	1000.00
	12	Indumati	600.00
	13	Shizuka	2000.00
	14	Suneo	400.00
	15	Ninja Hattori	3000.00

## 28. List payments made on a specific date

```
108 • SELECT *
109 FROM Payment
110 WHERE PaymentDate = '2023-10-25';
```

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

PaymentID	CourierID	LocationID	Amount	PaymentDate
NULL	NULL	NULL	NULL	NULL



## 29. Get Courier Information for Each Payment

```

311
312 • SELECT Payment.*, Courier.*
313 FROM Payment
314 LEFT JOIN Courier ON Payment.CourierID = Courier.CourierID;
315

```

Result Grid

Filter Rows:




Export:

Wrap Cell Content:

	PaymentID	CourierID	LocationID	Amount	PaymentDate	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	UserID	ServiceID	OrderID
▶	1	1	1	100.00	2023-10-24	1	Chhota Bheem	Dholakpur Village	Shinchan Nohara	Kasukabe, Japan	2.50	In Transit	TRK123456	2023-10-25	1	1	2023-10-25
	2	2	2	250.00	2023-10-19	2	Doraemon	Tokyo, Japan	Ninja Hattori	Shinobi Village	1.80	Delivered	TRK123457	2023-10-20	3	2	2023-10-20
	3	3	3	500.00	2023-10-29	3	Heidi	Swiss Alps	Jackie Chan	Hong Kong	3.00	Pending	TRK123458	2023-10-30	6	3	2023-10-30
	4	4	4	700.00	2023-10-26	4	Raju	Dholakpur Village	Chutki	Dholakpur Village	5.00	In Transit	TRK123459	2023-10-26	7	4	2023-10-26
	5	5	5	1500.00	2023-10-22	5	Kalia	Dholakpur Village	Indumati	Dholakpur Village	4.20	Delivered	TRK123460	2023-10-22	9	5	2023-10-22
	6	6	6	1200.00	2023-10-31	6	Nobita Nobi	Tokyo, Japan	Shizuka Minamoto	Tokyo, Japan	1.50	Pending	TRK123461	2023-10-31	11	6	2023-10-31
	7	7	7	800.00	2023-10-27	7	Gian	Tokyo, Japan	Suneo Honekawa	Tokyo, Japan	6.00	In Transit	TRK123462	2023-10-27	13	7	2023-10-27
	8	8	8	900.00	2023-10-23	8	Dekisugi	Tokyo, Japan	Nobita Nobi	Tokyo, Japan	2.00	Delivered	TRK123463	2023-10-23	15	8	2023-10-23
	9	9	9	300.00	2023-11-01	9	Shinchan Nohara	Kasukabe, Japan	Doraemon	Tokyo, Japan	3.50	Pending	TRK123464	2023-11-01	2	9	2023-11-01
	10	10	10	200.00	2023-10-28	10	Jackie Chan	Hong Kong	Heidi	Swiss Alps	7.00	In Transit	TRK123465	2023-10-28	5	10	2023-10-28
	11	11	11	1000.00	2023-10-24	11	Chutki	Dholakpur Village	Raju	Dholakpur Village	1.00	Delivered	TRK123466	2023-10-24	8	11	2023-10-24
	12	12	12	600.00	2023-11-02	12	Indumati	Dholakpur Village	Kalia	Dholakpur Village	2.30	Pending	TRK123467	2023-11-02	10	12	2023-11-02
	13	13	13	2000.00	2023-10-29	13	Shizuka	Tokyo, Japan	Gian	Tokyo, Japan	4.70	In Transit	TRK123468	2023-10-29	12	13	2023-10-29
	14	14	14	400.00	2023-10-21	14	Suneo	Tokyo, Japan	Dekisugi	Tokyo, Japan	3.80	Delivered	TRK123469	2023-10-21	14	14	2023-10-21
	15	15	15	3000.00	2023-11-03	15	Ninja Hattori	Shinobi Village	Chhota Bheem	Dholakpur Village	5.50	Pending	TRK123470	2023-11-03	4	15	2023-11-03

## 30. Get Payment Details with Location



```
315
316 • SELECT Payment.*, Location.*
317 FROM Payment
318 LEFT JOIN Location ON Payment.LocationID = Location.LocationID;
319
```

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

	PaymentID	CourierID	LocationID	Amount	PaymentDate	LocationID	LocationName	Address
1	1	1	1	100.00	2023-10-24	1	Dholakpur Branch	Dholakpur Village, India
2	2	2	2	250.00	2023-10-19	2	Kasukabe Branch	Kasukabe, Japan
3	3	3	3	500.00	2023-10-29	3	Tokyo Branch	Tokyo, Japan
4	4	4	4	700.00	2023-10-26	4	Hong Kong Branch	Hong Kong
5	5	5	5	1500.00	2023-10-22	5	Swiss Alps Branch	Swiss Alps
6	6	6	6	1200.00	2023-10-31	6	Mumbai Branch	Mumbai, India
7	7	7	7	800.00	2023-10-27	7	Delhi Branch	Delhi, India
8	8	8	8	900.00	2023-10-23	8	Bangalore Branch	Bangalore, India
9	9	9	9	300.00	2023-11-01	9	Chennai Branch	Chennai, India
10	10	10	10	200.00	2023-10-28	10	Kolkata Branch	Kolkata, India
11	11	11	11	1000.00	2023-10-24	11	New York Branch	New York, USA
12	12	12	12	600.00	2023-11-02	12	London Branch	London, UK
13	13	13	13	2000.00	2023-10-29	13	Paris Branch	Paris, France
14	14	14	14	400.00	2023-10-21	14	Sydney Branch	Sydney, Australia
15	15	15	15	3000.00	2023-11-03	15	Dubai Branch	Dubai, UAE

## 31. Calculating Total Payments for Each Courier

```
319
320 • SELECT Courier.CourierID, Courier.SenderName, SUM(Payment.Amount) AS TotalPayments
321 FROM Payment
322 INNER JOIN Courier ON Payment.CourierID = Courier.CourierID
323 GROUP BY Courier.CourierID;
324
```

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

	CourierID	SenderName	TotalPayments
▶	1	Chhota Bheem	100.00
	2	Doraemon	250.00
	3	Heidi	500.00
	4	Raju	700.00
	5	Kalia	1500.00
	6	Nobita Nobi	1200.00
	7	Gian	800.00
	8	Dekisugi	900.00
	9	Shinchan Nohara	300.00
	10	Jackie Chan	200.00
	11	Chutki	1000.00
	12	Indumati	600.00
	13	Shizuka	2000.00
	14	Suneo	400.00
	15	Ninja Hattori	3000.00

## 32. List Payments Within a Date Range

```
325 SELECT *
326 FROM Payment
327 WHERE PaymentDate BETWEEN '2023-10-20' AND '2023-10-30';
328
```

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

	PaymentID	CourierID	LocationID	Amount	PaymentDate
	1	1	1	100.00	2023-10-24
	3	3	3	500.00	2023-10-29
	4	4	4	700.00	2023-10-26
	5	5	5	1500.00	2023-10-22
	7	7	7	800.00	2023-10-27
	8	8	8	900.00	2023-10-23
	10	10	10	200.00	2023-10-28
	11	11	11	1000.00	2023-10-24
	13	13	13	2000.00	2023-10-29
	14	14	14	400.00	2023-10-21
	NULL	NULL	NULL	NULL	NULL



### 33. Retrieve a list of all users and their corresponding courier records, including cases where there are no matches on either side

```

328
329 • SELECT User.*, Courier.*
330 FROM User
331 LEFT JOIN Courier ON User.UserID = Courier.UserID;
332

```

UserID	Name	Email	Password	ContactNumber	Address	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber
1	Chhota Bheem	bheem@dholakpur.com	bheem123	9876543210	Dholakpur Village	1	Chhota Bheem	Dholakpur Village	Shinchan Nohara	Kasukabe, Japan	2.50	In Transit	TRK123456
2	Shinchan Nohara	shinchan@kasukabe.com	shinchan123	9876543211	Kasukabe, Japan	9	Shinchan Nohara	Kasukabe, Japan	Doraemon	Tokyo, Japan	3.50	Pending	TRK123464
3	Doraemon	doraemon@future.com	doraemon123	9876543212	Tokyo, Japan	2	Doraemon	Tokyo, Japan	Ninja Hattori	Shinobi Village	1.80	Delivered	TRK123457
4	Ninja Hattori	hattori@ninja.com	hattori123	9876543213	Shinobi Village	15	Ninja Hattori	Shinobi Village	Chhota Bheem	Dholakpur Village	5.50	Pending	TRK123470
5	Jackie Chan	jackie@kungfu.com	jackie123	9876543214	Hong Kong	10	Jackie Chan	Hong Kong	Heidi	Swiss Alps	7.00	In Transit	TRK123465
6	Heidi	heidi@alps.com	heidi123	9876543215	Swiss Alps	3	Heidi	Swiss Alps	Jackie Chan	Hong Kong	3.00	Pending	TRK123458
7	Raju	raju@dholakpur.com	raju123	9876543216	Dholakpur Village	4	Raju	Dholakpur Village	Chutki	Dholakpur Village	5.00	In Transit	TRK123459
8	Chutki	chutki@dholakpur.com	chutki123	9876543217	Dholakpur Village	11	Chutki	Dholakpur Village	Raju	Dholakpur Village	1.00	Delivered	TRK123466
9	Kalia	kalia@dholakpur.com	kalia123	9876543218	Dholakpur Village	5	Kalia	Dholakpur Village	Indumati	Dholakpur Village	4.20	Delivered	TRK123460
10	Indumati	indu@dholakpur.com	indu123	9876543219	Dholakpur Village	12	Indumati	Dholakpur Village	Kalia	Dholakpur Village	2.30	Pending	TRK123467
11	Nobita Nobi	nobita@future.com	nobita123	9876543220	Tokyo, Japan	6	Nobita Nobi	Tokyo, Japan	Shizuka Minamoto	Tokyo, Japan	1.50	Pending	TRK123461
12	Shizuka	shizuka@future.com	shizuka123	9876543221	Tokyo, Japan	13	Shizuka	Tokyo, Japan	Gian	Tokyo, Japan	4.70	In Transit	TRK123468
13	Gian	gian@future.com	gian123	9876543222	Tokyo, Japan	7	Gian	Tokyo, Japan	Suneo Honekawa	Tokyo, Japan	6.00	In Transit	TRK123462
14	Suneo	suneo@future.com	suneo123	9876543223	Tokyo, Japan	14	Suneo	Tokyo, Japan	Dekisugi	Tokyo, Japan	3.80	Delivered	TRK123469
15	Dekisugi	dehisugi@future.com	dehisugi123	9876543224	Tokyo, Japan	8	Dekisugi	Tokyo, Japan	Nobita Nobi	Tokyo, Japan	2.00	Delivered	TRK123463

34. Retrieve a list of all couriers and their corresponding services, including cases where there are no matches on either side

```

332
333 • SELECT Courier.*, CourierServices.*
334 FROM Courier
335 LEFT JOIN CourierServices ON Courier.ServiceID = CourierServices.ServiceID;
336

```

Result Grid		Filter Rows:		Export:		Wrap Cell Content:												
	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	UserID	ServiceID	OrderDate	EmployeeID	ServiceID	ServiceName	Cost		
▶	1	Chhota Bheem	Dholakpur Village	Shinchan Nohara	Kasukabe, Japan	2.50	In Transit	TRK123456	2023-10-25	1	1	2023-10-23	1	1	Standard Delivery	100.00		
	2	Doraemon	Tokyo, Japan	Ninja Hattori	Shinobi Village	1.80	Delivered	TRK123457	2023-10-20	3	2	2023-10-18	2	2	Express Delivery	250.00		
	3	Heidi	Swiss Alps	Jackie Chan	Hong Kong	3.00	Pending	TRK123458	2023-10-30	6	3	2023-10-28	3	3	Overnight Delivery	500.00		
	4	Raju	Dholakpur Village	Chutki	Dholakpur Village	5.00	In Transit	TRK123459	2023-10-26	7	4	2023-10-24	NULL	4	Same-Day Delivery	700.00		
	5	Kalia	Dholakpur Village	Indumati	Dholakpur Village	4.20	Delivered	TRK123460	2023-10-22	9	5	2023-10-20	NULL	5	International Delivery	1500.00		
	6	Nobita Nobi	Tokyo, Japan	Shizuka Minamoto	Tokyo, Japan	1.50	Pending	TRK123461	2023-10-31	11	6	2023-10-29	NULL	6	Heavy Parcel Delivery	1200.00		
	7	Gian	Tokyo, Japan	Suneo Honekawa	Tokyo, Japan	6.00	In Transit	TRK123462	2023-10-27	13	7	2023-10-25	NULL	7	Fragile Item Delivery	800.00		
	8	Dekisugi	Tokyo, Japan	Nobita Nobi	Tokyo, Japan	2.00	Delivered	TRK123463	2023-10-23	15	8	2023-10-21	NULL	8	Medical Supply Delivery	900.00		
	9	Shinchan Nohara	Kasukabe, Japan	Doraemon	Tokyo, Japan	3.50	Pending	TRK123464	2023-11-01	2	9	2023-10-30	NULL	9	Food Delivery	300.00		
	10	Jackie Chan	Hong Kong	Heidi	Swiss Alps	7.00	In Transit	TRK123465	2023-10-28	5	10	2023-10-26	NULL	10	Document Delivery	200.00		
	11	Chutki	Dholakpur Village	Raju	Dholakpur Village	1.00	Delivered	TRK123466	2023-10-24	8	11	2023-10-22	NULL	11	Electronics Delivery	1000.00		
	12	Indumati	Dholakpur Village	Kalia	Dholakpur Village	2.30	Pending	TRK123467	2023-11-02	10	12	2023-10-31	NULL	12	Gift Delivery	600.00		
	13	Shizuka	Tokyo, Japan	Gian	Tokyo, Japan	4.70	In Transit	TRK123468	2023-10-29	12	13	2023-10-27	NULL	13	Bulk Delivery	2000.00		
	14	Suneo	Tokyo, Japan	Dekisugi	Tokyo, Japan	3.80	Delivered	TRK123469	2023-10-21	14	14	2023-10-19	NULL	14	Eco-Friendly Delivery	400.00		
	15	Ninja Hattori	Shinobi Village	Chhota Bheem	Dholakpur Village	5.50	Pending	TRK123470	2023-11-03	4	15	2023-11-01	NULL	15	VIP Delivery	3000.00		



35. Retrieve a list of all employees and their corresponding payments, including cases where there are no matches on either side

336

337 • `SELECT Employee.*, Payment.*`  
 338 `FROM Employee`  
 339 `LEFT JOIN Payment ON Employee.EmployeeID = Payment.CourierID;`  
 340

Result Grid											
Filter Rows: <input type="text"/> Export:  Wrap Cell Content:											
	EmployeeID	Name	Email	ContactNumber	Role	Salary	PaymentID	CourierID	LocationID	Amount	PaymentDate
▶	1	Raju	raju@dholakpur.com	9876543220	Delivery Boy	20000.00	1	1	1	100.00	2023-10-24
	2	Jaggu	jaggu@dholakpur.com	9876543221	Manager	50000.00	2	2	2	250.00	2023-10-19
	3	Kalia	kalia@dholakpur.com	9876543222	Warehouse Staff	15000.00	3	3	3	500.00	2023-10-29
	4	Indumati	indu@dholakpur.com	9876543223	Customer Support	18000.00	4	4	4	700.00	2023-10-26
	5	Chutki	chutki@dholakpur.com	9876543224	Delivery Boy	20000.00	5	5	5	1500.00	2023-10-22
	6	Nobita Nobita	nobita@future.com	9876543225	IT Support	25000.00	6	6	6	1200.00	2023-10-31
	7	Shizuka	shizuka@future.com	9876543226	HR Manager	40000.00	7	7	7	800.00	2023-10-27
	8	Gian	gian@future.com	9876543227	Security	22000.00	8	8	8	900.00	2023-10-23
	9	Suneo	suneo@future.com	9876543228	Accountant	30000.00	9	9	9	300.00	2023-11-01
	10	Dekisugi	dehisugi@future.com	9876543229	Developer	35000.00	10	10	10	200.00	2023-10-28
	11	Heidi	heidi@alps.com	9876543230	Delivery Boy	20000.00	11	11	11	1000.00	2023-10-24
	12	Jackie Chan	jackie@kungfu.com	9876543231	Security	22000.00	12	12	12	600.00	2023-11-02
	13	Ninja Hattori	hattori@ninja.com	9876543232	Delivery Boy	20000.00	13	13	13	2000.00	2023-10-29
	14	Doraemon	doraemon@future.com	9876543233	IT Support	25000.00	14	14	14	400.00	2023-10-21
	15	Shinchan ...	shinchan@kasukabe....	9876543234	Customer Support	18000.00	15	15	15	3000.00	2023-11-03

## 36. List all users and all courier services, showing all possible combinations.

```
340
341 • SELECT User.*, CourierServices.*
342 FROM User
343 CROSS JOIN CourierServices;
344
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:


	UserID	Name	Email	Password	ContactNumber	Address	ServiceID	ServiceName	Cost
▶	15	Dekisugi	dekisugi@future.com	dekisugi123	9876543224	Tokyo, Japan	1	Standard Delivery	100.00
	14	Suneo	suneo@future.com	suneo123	9876543223	Tokyo, Japan	1	Standard Delivery	100.00
	13	Gian	gian@future.com	gian123	9876543222	Tokyo, Japan	1	Standard Delivery	100.00
	12	Shizuka	shizuka@future.com	shizuka123	9876543221	Tokyo, Japan	1	Standard Delivery	100.00
	11	Nobita Nobi	nobita@future.com	nobita123	9876543220	Tokyo, Japan	1	Standard Delivery	100.00
	10	Indumati	indu@dholakpur.com	indu123	9876543219	Dholakpur Village	1	Standard Delivery	100.00
	9	Kalia	kalia@dholakpur.com	kalia123	9876543218	Dholakpur Village	1	Standard Delivery	100.00
	8	Chutki	chutki@dholakpur.com	chutki123	9876543217	Dholakpur Village	1	Standard Delivery	100.00
	7	Raju	raju@dholakpur.com	raju123	9876543216	Dholakpur Village	1	Standard Delivery	100.00
	6	Heidi	heidi@alps.com	heidi123	9876543215	Swiss Alps	1	Standard Delivery	100.00
	5	Jackie Chan	jackie@kungfu.com	jackie123	9876543214	Hong Kong	1	Standard Delivery	100.00
	4	Ninja Hattori	hattori@ninja.com	hattori123	9876543213	Shinobi Village	1	Standard Delivery	100.00
	3	Doraemon	doraemon@future.com	doraemon...	9876543212	Tokyo, Japan	1	Standard Delivery	100.00
	2	Shinchan ...	shinchan@kasukabe....	shinchan123	9876543211	Kasukabe, Japan	1	Standard Delivery	100.00
	1	Chhota Bh...	bheem@dholakpur.com	bheem123	9876543210	Dholakpur Village	1	Standard Delivery	100.00
	15	Dekisugi	dekisugi@future.com	dekisugi123	9876543224	Tokyo, Japan	2	Express Delivery	250.00
	14	Suneo	suneo@future.com	suneo123	9876543223	Tokyo, Japan	2	Express Delivery	250.00
	13	Gian	gian@future.com	gian123	9876543222	Tokyo, Japan	2	Express Delivery	250.00
	12	Shizuka	shizuka@future.com	shizuka123	9876543221	Tokyo, Japan	2	Express Delivery	250.00
	11	Nobita Nobi	nobita@future.com	nobita123	9876543220	Tokyo, Japan	2	Express Delivery	250.00

Result 51 x



### 37. List all employees and all locations, showing all possible combinations:

```
344
345 • SELECT Employee.*, Location.*
346 FROM Employee
347 CROSS JOIN Location;
348
```

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

	EmployeeID	Name	Email	ContactNumber	Role	Salary	LocationID	LocationName	Address
	15	Shinchan Nohara	shinchan@kasukabe.com	9876543234	Customer Support	18000.00	1	Dholakpur Branch	Dholakpur Village, India
	14	Doraemon	doraemon@future.com	9876543233	IT Support	25000.00	1	Dholakpur Branch	Dholakpur Village, India
	13	Ninja Hattori	hattori@ninja.com	9876543232	Delivery Boy	20000.00	1	Dholakpur Branch	Dholakpur Village, India
	12	Jackie Chan	jackie@kungfu.com	9876543231	Security	22000.00	1	Dholakpur Branch	Dholakpur Village, India
	11	Heidi	heidi@alps.com	9876543230	Delivery Boy	20000.00	1	Dholakpur Branch	Dholakpur Village, India
	10	Dekisugi	dehisugi@future.com	9876543229	Developer	35000.00	1	Dholakpur Branch	Dholakpur Village, India
	9	Suneo	suneo@future.com	9876543228	Accountant	30000.00	1	Dholakpur Branch	Dholakpur Village, India
	8	Gian	gian@future.com	9876543227	Security	22000.00	1	Dholakpur Branch	Dholakpur Village, India
	7	Shizuka	shizuka@future.com	9876543226	HR Manager	40000.00	1	Dholakpur Branch	Dholakpur Village, India
	6	Nobita Nobi	nobita@future.com	9876543225	IT Support	25000.00	1	Dholakpur Branch	Dholakpur Village, India
	5	Chutki	chutki@dholakpur.com	9876543224	Delivery Boy	20000.00	1	Dholakpur Branch	Dholakpur Village, India
	4	Indumati	indu@dholakpur.com	9876543223	Customer Support	18000.00	1	Dholakpur Branch	Dholakpur Village, India
	3	Kalia	kalia@dholakpur.com	9876543222	Warehouse Staff	15000.00	1	Dholakpur Branch	Dholakpur Village, India
	2	Jaggu	jaggu@dholakpur.com	9876543221	Manager	50000.00	1	Dholakpur Branch	Dholakpur Village, India
	1	Raju	raju@dholakpur.com	9876543220	Delivery Boy	20000.00	1	Dholakpur Branch	Dholakpur Village, India
	15	Shinchan Nohara	shinchan@kasukabe.com	9876543234	Customer Support	18000.00	2	Kasukabe Branch	Kasukabe, Japan
	14	Doraemon	doraemon@future.com	9876543233	IT Support	25000.00	2	Kasukabe Branch	Kasukabe, Japan
	13	Ninja Hattori	hattori@ninja.com	9876543232	Delivery Boy	20000.00	2	Kasukabe Branch	Kasukabe, Japan
	12	Jackie Chan	jackie@kungfu.com	9876543231	Security	22000.00	2	Kasukabe Branch	Kasukabe, Japan
	11	Heidi	heidi@alps.com	9876543230	Delivery Boy	20000.00	2	Kasukabe Branch	Kasukabe, Japan

Result 52 x

## 38. Retrieve a list of couriers and their corresponding sender information (if available)

348

349 • `SELECT Courier.*, User.Name AS SenderName, User.Address AS SenderAddress`

350 `FROM Courier`

351 `LEFT JOIN User ON Courier.UserID = User.UserID;`

352

Result Grid															
		Filter Rows:		Export:		Wrap Cell Content:									
	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	UserID	ServiceID	OrderDate	EmployeeID	SenderName	SenderAddress
▶	1	Chhota Bheem	Dholakpur Village	Shinchan Nohara	Kasukabe, Japan	2.50	In Transit	TRK123456	2023-10-25	1	1	2023-10-23	1	Chhota Bheem	Dholakpur Village
	2	Doraemon	Tokyo, Japan	Ninja Hattori	Shinobi Village	1.80	Delivered	TRK123457	2023-10-20	3	2	2023-10-18	2	Doraemon	Tokyo, Japan
	3	Heidi	Swiss Alps	Jackie Chan	Hong Kong	3.00	Pending	TRK123458	2023-10-30	6	3	2023-10-28	3	Heidi	Swiss Alps
	4	Raju	Dholakpur Village	Chutki	Dholakpur Village	5.00	In Transit	TRK123459	2023-10-26	7	4	2023-10-24	NULL	Raju	Dholakpur Village
	5	Kalia	Dholakpur Village	Indumati	Dholakpur Village	4.20	Delivered	TRK123460	2023-10-22	9	5	2023-10-20	NULL	Kalia	Dholakpur Village
	6	Nobita Nobi	Tokyo, Japan	Shizuka Minamoto	Tokyo, Japan	1.50	Pending	TRK123461	2023-10-31	11	6	2023-10-29	NULL	Nobita Nobi	Tokyo, Japan
	7	Gian	Tokyo, Japan	Suneo Honekawa	Tokyo, Japan	6.00	In Transit	TRK123462	2023-10-27	13	7	2023-10-25	NULL	Gian	Tokyo, Japan
	8	Dekisugi	Tokyo, Japan	Nobita Nobi	Tokyo, Japan	2.00	Delivered	TRK123463	2023-10-23	15	8	2023-10-21	NULL	Dekisugi	Tokyo, Japan
	9	Shinchan Nohara	Kasukabe, Japan	Doraemon	Tokyo, Japan	3.50	Pending	TRK123464	2023-11-01	2	9	2023-10-30	NULL	Shinchan Nohara	Kasukabe, Japan
	10	Jackie Chan	Hong Kong	Heidi	Swiss Alps	7.00	In Transit	TRK123465	2023-10-28	5	10	2023-10-26	NULL	Jackie Chan	Hong Kong
	11	Chutki	Dholakpur Village	Raju	Dholakpur Village	1.00	Delivered	TRK123466	2023-10-24	8	11	2023-10-22	NULL	Chutki	Dholakpur Village
	12	Indumati	Dholakpur Village	Kalia	Dholakpur Village	2.30	Pending	TRK123467	2023-11-02	10	12	2023-10-31	NULL	Indumati	Dholakpur Village
	13	Shizuka	Tokyo, Japan	Gian	Tokyo, Japan	4.70	In Transit	TRK123468	2023-10-29	12	13	2023-10-27	NULL	Shizuka	Tokyo, Japan
	14	Suneo	Tokyo, Japan	Dekisugi	Tokyo, Japan	3.80	Delivered	TRK123469	2023-10-21	14	14	2023-10-19	NULL	Suneo	Tokyo, Japan
	15	Ninja Hattori	Shinobi Village	Chhota Bheem	Dholakpur Village	5.50	Pending	TRK123470	2023-11-03	4	15	2023-11-01	NULL	Ninja Hattori	Shinobi Village



## 39. Retrieve a list of couriers and their corresponding receiver information (if available):

352  
353  
354  
355  
356

```
SELECT Courier.*, User.Name AS ReceiverName, User.Address AS ReceiverAddress
FROM Courier
LEFT JOIN User ON Courier.UserID = User.UserID;
```

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

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	UserID	ServiceID	OrderDate	EmployeeID	ReceiverName	ReceiverAddress
1	Chhota Bheem	Dholakpur Village	Shinchan Nohara	Kasukabe, Japan	2.50	In Transit	TRK123456	2023-10-25	1	1	2023-10-23	1	Chhota Bheem	Dholakpur Village
2	Doraemon	Tokyo, Japan	Ninja Hattori	Shinobi Village	1.80	Delivered	TRK123457	2023-10-20	3	2	2023-10-18	2	Doraemon	Tokyo, Japan
3	Heidi	Swiss Alps	Jackie Chan	Hong Kong	3.00	Pending	TRK123458	2023-10-30	6	3	2023-10-28	3	Heidi	Swiss Alps
4	Raju	Dholakpur Village	Chutki	Dholakpur Village	5.00	In Transit	TRK123459	2023-10-26	7	4	2023-10-24	NULL	Raju	Dholakpur Village
5	Kalia	Dholakpur Village	Indumati	Dholakpur Village	4.20	Delivered	TRK123460	2023-10-22	9	5	2023-10-20	NULL	Kalia	Dholakpur Village
6	Nobita Nobi	Tokyo, Japan	Shizuka Minamoto	Tokyo, Japan	1.50	Pending	TRK123461	2023-10-31	11	6	2023-10-29	NULL	Nobita Nobi	Tokyo, Japan
7	Gian	Tokyo, Japan	Suneo Honekawa	Tokyo, Japan	6.00	In Transit	TRK123462	2023-10-27	13	7	2023-10-25	NULL	Gian	Tokyo, Japan
8	Dekisugi	Tokyo, Japan	Nobita Nobi	Tokyo, Japan	2.00	Delivered	TRK123463	2023-10-23	15	8	2023-10-21	NULL	Dekisugi	Tokyo, Japan
9	Shinchan Nohara	Kasukabe, Japan	Doraemon	Tokyo, Japan	3.50	Pending	TRK123464	2023-11-01	2	9	2023-10-30	NULL	Shinchan Nohara	Kasukabe, Japan
10	Jackie Chan	Hong Kong	Heidi	Swiss Alps	7.00	In Transit	TRK123465	2023-10-28	5	10	2023-10-26	NULL	Jackie Chan	Hong Kong
11	Chutki	Dholakpur Village	Raju	Dholakpur Village	1.00	Delivered	TRK123466	2023-10-24	8	11	2023-10-22	NULL	Chutki	Dholakpur Village
12	Indumati	Dholakpur Village	Kalia	Dholakpur Village	2.30	Pending	TRK123467	2023-11-02	10	12	2023-10-31	NULL	Indumati	Dholakpur Village
13	Shizuka	Tokyo, Japan	Gian	Tokyo, Japan	4.70	In Transit	TRK123468	2023-10-29	12	13	2023-10-27	NULL	Shizuka	Tokyo, Japan
14	Suneo	Tokyo, Japan	Dekisugi	Tokyo, Japan	3.80	Delivered	TRK123469	2023-10-21	14	14	2023-10-19	NULL	Suneo	Tokyo, Japan
15	Ninja Hattori	Shinobi Village	Chhota Bheem	Dholakpur Village	5.50	Pending	TRK123470	2023-11-03	4	15	2023-11-01	NULL	Ninja Hattori	Shinobi Village

## 40. Retrieve a list of couriers along with the courier service details (if available):

```

356
357 • SELECT Courier.*, CourierServices.*
358 FROM Courier
359 LEFT JOIN CourierServices ON Courier.ServiceID = CourierServices.ServiceID;
360
361 • SELECT Employee.EmployeeID, Employee.Name, COUNT(Courier.CourierID) AS TotalCouriers

```

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	UserID	ServiceID	OrderDate	EmployeeID	ServiceID	ServiceName	Cost
1	Chhota Bheem	Dholakpur Village	Shinchan Nohara	Kasukabe, Japan	2.50	In Transit	TRK123456	2023-10-25	1	1	2023-10-23	1	1	Standard Delivery	100.00
2	Doraemon	Tokyo, Japan	Ninja Hattori	Shinobi Village	1.80	Delivered	TRK123457	2023-10-20	3	2	2023-10-18	2	2	Express Delivery	250.00
3	Heidi	Swiss Alps	Jackie Chan	Hong Kong	3.00	Pending	TRK123458	2023-10-30	6	3	2023-10-28	3	3	Overnight Delivery	500.00
4	Raju	Dholakpur Village	Chutki	Dholakpur Village	5.00	In Transit	TRK123459	2023-10-26	7	4	2023-10-24	NULL	4	Same-Day Delivery	700.00
5	Kalia	Dholakpur Village	Indumati	Dholakpur Village	4.20	Delivered	TRK123460	2023-10-22	9	5	2023-10-20	NULL	5	International Delivery	1500.00
6	Nobita Nobi	Tokyo, Japan	Shizuka Minamoto	Tokyo, Japan	1.50	Pending	TRK123461	2023-10-31	11	6	2023-10-29	NULL	6	Heavy Parcel Delivery	1200.00
7	Gian	Tokyo, Japan	Suneo Honekawa	Tokyo, Japan	6.00	In Transit	TRK123462	2023-10-27	13	7	2023-10-25	NULL	7	Fragile Item Delivery	800.00
8	Dekisugi	Tokyo, Japan	Nobita Nobi	Tokyo, Japan	2.00	Delivered	TRK123463	2023-10-23	15	8	2023-10-21	NULL	8	Medical Supply Delivery	900.00
9	Shinchan Nohara	Kasukabe, Japan	Doraemon	Tokyo, Japan	3.50	Pending	TRK123464	2023-11-01	2	9	2023-10-30	NULL	9	Food Delivery	300.00
10	Jackie Chan	Hong Kong	Heidi	Swiss Alps	7.00	In Transit	TRK123465	2023-10-28	5	10	2023-10-26	NULL	10	Document Delivery	200.00
11	Chutki	Dholakpur Village	Raju	Dholakpur Village	1.00	Delivered	TRK123466	2023-10-24	8	11	2023-10-22	NULL	11	Electronics Delivery	1000.00
12	Indumati	Dholakpur Village	Kalia	Dholakpur Village	2.30	Pending	TRK123467	2023-11-02	10	12	2023-10-31	NULL	12	Gift Delivery	600.00
13	Shizuka	Tokyo, Japan	Gian	Tokyo, Japan	4.70	In Transit	TRK123468	2023-10-29	12	13	2023-10-27	NULL	13	Bulk Delivery	2000.00
14	Suneo	Tokyo, Japan	Dekisugi	Tokyo, Japan	3.80	Delivered	TRK123469	2023-10-21	14	14	2023-10-19	NULL	14	Eco-Friendly Delivery	400.00
15	Ninja Hattori	Shinobi Village	Chhota Bheem	Dholakpur Village	5.50	Pending	TRK123470	2023-11-03	4	15	2023-11-01	NULL	15	VIP Delivery	3000.00



41. Retrieve a list of employees and the number of couriers assigned to each employee:

```
360  
361 • SELECT Employee.EmployeeID, Employee.Name, COUNT(Courier.CourierID) AS TotalCouriers  
362 FROM Employee  
363 LEFT JOIN Courier ON Employee.EmployeeID = Courier.EmployeeID  
364 GROUP BY Employee.EmployeeID;  
365
```

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

	EmployeeID	Name	TotalCouriers
▶	1	Raju	1
	2	Jaggu	1
	3	Kalia	1
	4	Indumati	0
	5	Chutki	0
	6	Nobita Nobi	0
	7	Shizuka	0
	8	Gian	0
	9	Suneo	0
	10	Dekisugi	0
	11	Heidi	0
	12	Jackie Chan	0
	13	Ninja Hattori	0
	14	Doraemon	0
	15	Shinchan ...	0

42. Retrieve a list of locations and the total payment amount received at each location:

365



366 • `SELECT Location.LocationID, Location.LocationName, SUM(Payment.Amount) AS TotalPayments`

367 `FROM Location`

368 `LEFT JOIN Payment ON Location.LocationID = Payment.LocationID`

369 `GROUP BY Location.LocationID;`

370

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

	LocationID	LocationName	TotalPayments
▶	1	Dholakpur Branch	100.00
	2	Kasukabe Branch	250.00
	3	Tokyo Branch	500.00
	4	Hong Kong Branch	700.00
	5	Swiss Alps Branch	1500.00
	6	Mumbai Branch	1200.00
	7	Delhi Branch	800.00
	8	Bangalore Branch	900.00
	9	Chennai Branch	300.00
	10	Kolkata Branch	200.00
	11	New York Branch	1000.00
	12	London Branch	600.00
	13	Paris Branch	2000.00
	14	Sydney Branch	400.00
	15	Dubai Branch	3000.00

43. Retrieve all couriers sent by the same sender (based on SenderName).

```
SELECT *
FROM Courier
WHERE SenderName = 'Chhota Bheem';
```

[illegible]

44. List all employees who share the same role.

```
374  
375 • SELECT Role, COUNT(*) AS TotalEmployees  
376 FROM Employee  
377 GROUP BY Role  
378 HAVING TotalEmployees > 1;  
379
```

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

	Role	TotalEmployees
▶	Delivery Boy	4
	Customer Support	2
	IT Support	2
	Security	2

45. Retrieve all payments made for couriers sent from the same location.

379  
380  
381  
382  
383

```
SELECT *  
FROM Payment  
WHERE LocationID = 1;
```

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

	PaymentID	CourierID	LocationID	Amount	PaymentDate
▶	1	1	1	100.00	2023-10-24
•	NULL	NULL	NULL	NULL	NULL

46. Retrieve all couriers sent from the same location (based on SenderAddress).

```
SELECT *  
FROM Courier  
WHERE SenderAddress = 'Dholakpur Village';
```

[illegible]



## 47. List employees and the number of couriers they have delivered:

```
387
388 • SELECT Employee.EmployeeID, Employee.Name, COUNT(Courier.CourierID) AS TotalCouriersDelivered
389 FROM Employee
390 LEFT JOIN Courier ON Employee.EmployeeID = Courier.EmployeeID
391 GROUP BY Employee.EmployeeID;
392
```

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

	EmployeeID	Name	TotalCouriersDelivered
▶	1	Raju	1
	2	Jaggu	1
	3	Kalia	1
	4	Indumati	0
	5	Chutki	0
	6	Nobita Nobi	0
	7	Shizuka	0
	8	Gian	0
	9	Suneo	0
	10	Dekisugi	0
	11	Heidi	0
	12	Jackie Chan	0
	13	Ninja Hattori	0
	14	Doraemon	0
	15	Shinchan ...	0

48. Find couriers that were paid an amount greater than the cost of their respective courier services

392

393 •

```
SELECT Courier.*
```

394

```
FROM Courier
```

395

```
INNER JOIN Payment ON Courier.CourierID = Payment.CourierID
```

396

```
INNER JOIN CourierServices ON Courier.ServiceID = CourierServices.ServiceID
```

397

```
WHERE Payment.Amount > CourierServices.Cost;
```

398

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

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	UserID	ServiceID	OrderDate	EmployeeID
-----------	------------	---------------	--------------	-----------------	--------	--------	----------------	--------------	--------	-----------	-----------	------------

Scope: Inner Queries, Non Equi Joins, Equi joins, Exist, Any, All  
49. Find couriers that have a weight greater than the average weight of all couriers

```
SELECT *  
FROM Courier  
WHERE Weight > (SELECT AVG(Weight) FROM Courier);
```

[illegible]

50. Find the names of all employees who have a salary greater than the average salary:




```
402
403 • SELECT Name
404     FROM Employee
405     WHERE Salary > (SELECT AVG(Salary) FROM Employee);
406
```

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

	Name
▶	Jaggu
	Shizuka
	Suneo
	Dekisugi

51. Find the total cost of all courier services where the cost is less than the maximum cost

```
406  
407 • SELECT SUM(Cost) AS TotalCost  
408 FROM CourierServices  
409 WHERE Cost < (SELECT MAX(Cost) FROM CourierServices);  
410
```

Result Grid  Filter Rows:  Export:  Wrap Cell Content: 

TotalCost
▶ 10450.00





53. Find the locations where the maximum payment amount was made

```
414
415 • SELECT LocationID, LocationName
416 FROM Location
417 WHERE LocationID = (
418     SELECT LocationID
419     FROM Payment
420     WHERE Amount = (SELECT MAX(Amount) FROM Payment)
421 );
422
```

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	LocationID	LocationName
15	Dubai Branch	
NULL	NULL	

54. Find all couriers whose weight is greater than the weight of all couriers sent by a specific sender (e.g., 'SenderName'):

```
423 SELECT *
424 FROM Courier
425 WHERE Weight > ALL (
426     SELECT Weight
427     FROM Courier
428     WHERE SenderName = 'Chhota Bheem'
429 );
```

[illegible]