

## Task1 Database Design

```
CREATE TABLE User (
    UserID INT PRIMARY KEY,
    Name VARCHAR(255),
    Email VARCHAR(255) UNIQUE,
    Password VARCHAR(255),
    ContactNumber VARCHAR(20),
    Address TEXT
);
```

```
CREATE TABLE 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
);
```

```
CREATE TABLE CourierServices (
    ServiceID INT PRIMARY KEY,
    ServiceName VARCHAR(100),
    Cost DECIMAL(8, 2)
);
```

```
CREATE TABLE Employee (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(255),
    Email VARCHAR(255) UNIQUE,
```

```
ContactNumber VARCHAR(20),  
Role VARCHAR(50),  
Salary DECIMAL(10, 2)  
);
```

```
CREATE TABLE Location (  
    LocationID INT PRIMARY KEY,  
    LocationName VARCHAR(100),  
    Address TEXT  
);
```

```
CREATE TABLE Payment (  
    PaymentID INT PRIMARY KEY,  
    CourierID INT,  
    LocationID INT,  
    Amount DECIMAL(10, 2),  
    PaymentDate DATE,  
    FOREIGN KEY (CourierID) REFERENCES Courier(CourierID),  
    FOREIGN KEY (LocationID) REFERENCES Location(LocationID)  
);
```

```
INSERT INTO User (UserID, Name, Email, Password, ContactNumber, Address)  
VALUES  
(1, 'John Doe', 'john@example.com', 'password123', '1234567890', '123 Main St'),  
(2, 'Jane Smith', 'jane@example.com', 'password456', '0987654321', '456 Elm St'),  
(3, 'Alice Johnson', 'alice@example.com', 'password789', '9876543210', '789 Oak St'),  
(4, 'Bob Brown', 'bob@example.com', 'passwordabc', '4561237890', '321 Maple St'),  
(5, 'Eve Wilson', 'eve@example.com', 'passworddefg', '7894561230', '654 Pine St');
```

```
INSERT INTO Courier (CourierID, SenderName, SenderAddress, ReceiverName, ReceiverAddress,  
Weight, Status, TrackingNumber, DeliveryDate)  
VALUES
```

```
(1, 'John Doe', '123 Main St', 'Jane Smith', '456 Elm St', 3.5, 'In transit', 'ABC123', '2024-04-20'),  
(2, 'Alice Johnson', '789 Oak St', 'Bob Brown', '321 Maple St', 2.0, 'Delivered', 'DEF456', '2024-04-18'),  
(3, 'Eve Wilson', '654 Pine St', 'John Doe', '123 Main St', 1.8, 'Pending', 'GHI789', NULL),  
(4, 'Bob Brown', '321 Maple St', 'Alice Johnson', '789 Oak St', 4.2, 'In transit', 'JKL012', '2024-04-22'),  
(5, 'Jane Smith', '456 Elm St', 'Eve Wilson', '654 Pine St', 5.0, 'Pending', 'MNO345', NULL);
```

```
INSERT INTO CourierServices (ServiceID, ServiceName, Cost)
```

```
VALUES
```

```
(1, 'Standard', 10.00),  
(2, 'Express', 20.00),  
(3, 'Same Day', 30.00),  
(4, 'Overnight', 15.00),  
(5, 'International', 50.00);
```

```
INSERT INTO Employee (EmployeeID, Name, Email, ContactNumber, Role, Salary)
```

```
VALUES
```

```
(1, 'Mark Johnson', 'mark@example.com', '1239876540', 'Manager', 50000.00),  
(2, 'Emily Brown', 'emily@example.com', '9876543210', 'Clerk', 30000.00),  
(3, 'Michael Smith', 'michael@example.com', '4567890123', 'Driver', 35000.00),  
(4, 'Jessica Wilson', 'jessica@example.com', '3216549870', 'Administrator', 40000.00),  
(5, 'David Davis', 'david@example.com', '6541237890', 'Courier', 32000.00);
```

```
INSERT INTO Location (LocationID, LocationName, Address)
```

```
VALUES
```

```
(1, 'Office 1', '123 Main St'),  
(2, 'Office 2', '456 Elm St'),  
(3, 'Office 3', '789 Oak St'),  
(4, 'Office 4', '321 Maple St'),  
(5, 'Office 5', '654 Pine St');
```

```
INSERT INTO Payment (PaymentID, CourierID, LocationID, Amount, PaymentDate)
VALUES
(1, 1, 2, 15.00, '2024-04-18'),
(2, 2, 3, 20.00, '2024-04-19'),
(3, 3, 1, 25.00, '2024-04-20'),
(4, 4, 2, 18.00, '2024-04-21'),
(5, 5, 3, 22.00, '2024-04-22');
```

```
ALTER TABLE courier
ADD COLUMN ServiceID INT,
ADD CONSTRAINT ServiceID FOREIGN KEY (ServiceID) REFERENCES
CourierServices(ServiceID);
```

```
UPDATE Courier
SET ServiceID =
CASE
    WHEN CourierID = 1 THEN 1
    WHEN CourierID = 2 THEN 2
    WHEN CourierID = 3 THEN 3
    WHEN CourierID = 4 THEN 4
    WHEN CourierID = 5 THEN 5
    ELSE NULL
END
WHERE CourierID > 0;
```

### 1) Courier Table

The screenshot shows a database result grid titled "Result Grid". The toolbar includes icons for Filter Rows, Edit, Export/Import, and Wrap Cell Content. The table has columns: CourierID, SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate, EmployeeID, and ServiceID. The data consists of 6 rows, with the last row being a placeholder row indicated by an asterisk (\*).

	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	EmployeeID	ServiceID
▶	1	John Doe	123 Main St	Jane Smith	456 Elm St	3.50	In transit	ABC123	2024-04-20	1	1
▶	2	Alice Johnson	789 Oak St	Bob Brown	321 Maple St	2.00	Delivered	DEF456	2024-04-18	2	2
▶	3	Eve Wilson	654 Pine St	John Doe	123 Main St	1.80	Pending	GHI789	NULL	1	3
▶	4	Bob Brown	321 Maple St	Alice Johnson	789 Oak St	4.20	In transit	JKL012	2024-04-22	3	4
*	5	Jane Smith	456 Elm St	Eve Wilson	654 Pine St	5.00	Pending	MNO345	NULL	2	5
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

### 2) Courier Services Table

The screenshot shows a database result grid titled "Result Grid". The toolbar includes icons for Filter Rows, Edit, Export/Import, and Wrap Cell Content. The table has columns: ServiceID, ServiceName, and Cost. The data consists of 6 rows, with the last row being a placeholder row indicated by an asterisk (\*).

	ServiceID	ServiceName	Cost
▶	1	Standard	10.00
▶	2	Express	20.00
▶	3	Same Day	30.00
▶	4	Overnight	15.00
*	5	International	50.00
*	HULL	HULL	HULL

### 3) Employee Table

The screenshot shows a database result grid titled "Result Grid". The toolbar includes icons for Filter Rows, Edit, Export/Import, and Wrap Cell Content. The table has columns: EmployeeID, Name, Email, ContactNumber, Role, and Salary. The data consists of 6 rows, with the last row being a placeholder row indicated by an asterisk (\*).

	EmployeeID	Name	Email	ContactNumber	Role	Salary
▶	1	Mark Johnson	mark@example.com	1239876540	Manager	50000.00
▶	2	Emily Brown	emily@example.com	9876543210	Clerk	30000.00
▶	3	Michael Smith	michael@example.com	4567890123	Driver	35000.00
▶	4	Jessica Wilson	jessica@example.com	3216549870	Administrator	40000.00
▶	5	David Davis	david@example.com	6541237890	Courier	32000.00
*	HULL	HULL	HULL	HULL	HULL	HULL

#### 4) Location Table

A screenshot of a database application showing the 'Result Grid' for the Location Table. The grid has columns for LocationID, LocationName, and Address. There are 6 rows of data, indexed from 1 to 5, plus a header row and a footer row marked with an asterisk (\*). The data shows five office locations with their addresses.

	LocationID	LocationName	Address
1	1	Office 1	123 Main St
2	2	Office 2	456 Elm St
3	3	Office 3	789 Oak St
4	4	Office 4	321 Maple St
5	5	Office 5	654 Pine St
*	HULL	HULL	HULL

#### 5) Payment Table

A screenshot of a database application showing the 'Result Grid' for the Payment Table. The grid has columns for PaymentID, CourierID, LocationID, Amount, and PaymentDate. There are 6 rows of data, indexed from 1 to 5, plus a header row and a footer row marked with an asterisk (\*). The data shows five payments made by different couriers to various locations.

	PaymentID	CourierID	LocationID	Amount	PaymentDate
1	1	1	2	15.00	2024-04-18
2	2	2	3	20.00	2024-04-19
3	3	3	1	25.00	2024-04-20
4	4	4	2	18.00	2024-04-21
5	5	5	3	22.00	2024-04-22
*	HULL	HULL	HULL	HULL	HULL

#### 6) User Table

A screenshot of a database application showing the 'Result Grid' for the User Table. The grid has columns for UserID, Name, Email, Password, ContactNumber, and Address. There are 6 rows of data, indexed from 1 to 5, plus a header row and a footer row marked with an asterisk (\*). The data shows five user accounts with their contact information.

	UserID	Name	Email	Password	ContactNumber	Address
1	1	John Doe	john@example.com	password123	1234567890	123 Main St
2	2	Jane Smith	jane@example.com	password456	0987654321	456 Elm St
3	3	Alice Johnson	alice@example.com	password789	9876543210	789 Oak St
4	4	Bob Brown	bob@example.com	passwordabc	4561237890	321 Maple St
5	5	Eve Wilson	eve@example.com	passworddefg	7894561230	654 Pine St
*	HULL	HULL	HULL	HULL	HULL	HULL

-- 1. List all customers:

SELECT \*

FROM user;

The screenshot shows the MySQL Workbench interface. On the left, the Navigator pane displays the schema structure of the 'assignment1' database, including tables like courier, courierservices, employee, payment, location, and user, as well as views, stored procedures, functions, and system tables like college and sys. The main workspace is titled 'Database Creation SQL File 4\*' and contains the following SQL code:

```
1 -- 1. List all customers:  
2 • SELECT *  
3 FROM user;
```

Below the code, there is a 'Result Grid' showing the data from the 'user' table:

User ID	Name	Email	Password	Contact Number	Address
1	John Doe	john@example.com	password123	1234567890	123 Main St
2	Jane Smith	jane@example.com	password456	0987654321	456 Elm St
3	Alice Johnson	alice@example.com	password789	9876543210	789 Oak St
4	Bob Brown	bob@example.com	passwordabc	4561237890	321 Maple St
5	Eve Wilson	eve@example.com	passworddefg	7894561230	654 Pine St
*	HULL	HULL	HULL	HULL	HULL

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

SELECT \*

FROM Courier

```
WHERE SenderName = 'John Doe';
```

### -- 3. List all couriers

```
SELECT *\nFROM Courier;
```

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

```
SELECT *\nFROM Courier\nWHERE CourierID = 5;
```

-- 5. List all deliveries for a specific courier:

```
SELECT *  
FROM Courier  
WHERE SenderName = 'Bob Brown';
```

The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator and Schemas for the 'assignment1' database, listing tables like 'courier', 'courierservices', 'employee', etc. The right pane shows the 'SQL File 4\*' window with the following query:

```
-- 5. List all deliveries for a specific courier:  
SELECT *  
FROM Courier  
WHERE SenderName = 'Bob Brown';
```

The result grid shows one row of data:

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
4	Bob Brown	321 Maple St	Alice Johnson	789 Oak St	4.20	In transit	JKL012	2024-04-01

-- 6. List all undelivered packages:

```
SELECT *  
FROM Courier  
WHERE Status IN ('In transit', 'Pending');
```

The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator and Schemas for the 'assignment1' database. The right pane shows the 'SQL File 4\*' window with the following query:

```
-- 6. List all undelivered packages:  
SELECT *  
FROM Courier  
WHERE Status IN ('In transit', 'Pending');
```

The result grid shows four rows of data:

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
1	John Doe	123 Main St	Jane Smith	456 Elm St	3.50	In transit	ABC123	2024-04-01
3	Eve Wilson	654 Pine St	John Doe	123 Main St	1.80	Pending	GHI789	NULL
4	Bob Brown	321 Maple St	Alice Johnson	789 Oak St	4.20	In transit	JKL012	2024-04-01
5	Jane Smith	456 Elm St	Eve Wilson	654 Pine St	5.00	Pending	MNO345	NULL

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

```
SELECT *  
FROM Courier  
WHERE DeliveryDate = current_date();
```

The screenshot shows the SQL Server Management Studio interface. On the left is a tree view of database objects under the schema 'assignment1'. The central pane contains the following SQL code:

```
28  
29 -- 7. List all packages that are scheduled for delivery today:  
30  
31 • SELECT *  
32 FROM Courier  
33 WHERE DeliveryDate = current_date();  
34  
35
```

Below the code is a 'Result Grid' showing the results of the query. The columns are: CourierID, SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate, EmployeeID, and ServiceID. The data returned is:

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	EmployeeID	ServiceID
4	Bob Brown	321 Maple St	Alice Johnson	789 Oak St	4.20	In transit	JKL012	2024-04-22	3	4
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

-- 8. List all packages with a specific status:

```
SELECT *  
FROM courier  
WHERE Status= 'In transit';
```

The screenshot shows the SQL Server Management Studio interface. On the left is a tree view of database objects under the schema 'assignment1'. The central pane contains the following SQL code:

```
35 -- 8. List all packages with a specific status:  
36  
37 • SELECT *  
38 FROM courier  
39 WHERE Status= 'In transit';  
40  
41  
42  
43
```

Below the code is a 'Result Grid' showing the results of the query. The columns are: CourierID, SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, and DeliveryDate. The data returned is:

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
1	John Doe	123 Main St	Jane Smith	456 Elm St	3.50	In transit	ABC123	2024-04-20
4	Bob Brown	321 Maple St	Alice Johnson	789 Oak St	4.20	In transit	JKL012	2024-04-22
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

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

```
SELECT SenderName, COUNT(*) AS TotalPackages  
FROM Courier  
GROUP BY SenderName;
```

The screenshot shows the SQL Server Management Studio interface. On the left is a tree view of database objects under the schema 'assignment1'. The central pane contains the following SQL code:

```
40  
41 -- 9 Calculate the total number of packages for each courier.  
42 • SELECT SenderName, COUNT(*) AS TotalPackages  
43 FROM Courier  
44 GROUP BY SenderName;  
45  
46  
47  
48  
49  
50  
51
```

Below the code is a 'Result Grid' showing the results of the query. The columns are: SenderName and TotalPackages. The data returned is:

SenderName	TotalPackages
John Doe	1
Alice Johnson	1
Eve Wilson	1
Bob Brown	1
Jane Smith	1

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

```
SELECT SenderName,  
       AVG(DATEDIFF(DeliveryDate, PaymentDate)) AS AverageDeliveryTime  
  FROM Courier  
 JOIN Payment ON Courier.CourierID = Payment.CourierID  
 GROUP BY SenderName;
```

The screenshot shows the SQL Server Management Studio interface. The left pane displays the database schema with tables like 'courier', 'payment', and 'location'. The main pane shows the SQL code for query 10, which calculates the average delivery time for each courier. The results grid below shows the output:

SenderName	AverageDeliveryTime
John Doe	2.0000
Alice Johnson	-1.0000
Eve Wilson	NULL
Bob Brown	1.0000
Jane Smith	NULL

-- 11. List all packages with a specific weight range:

```
SELECT *  
  FROM courier  
 WHERE Weight >= 2;
```

The screenshot shows the SQL Server Management Studio interface. The left pane displays the database schema with tables like 'courier', 'payment', and 'location'. The main pane shows the SQL code for query 11, which lists packages with a weight of 2 or more. The results grid below shows the output:

CounterID	SenderId	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
1	John Doe	123 Main St	Jane Smith	456 Elm St	3.50	In transit	ABC123	2024-04-20
2	Alice Johnson	789 Oak St	Bob Brown	321 Maple St	2.00	Delivered	DEF456	2024-04-18
4	Bob Brown	321 Maple St	Alice Johnson	789 Oak St	4.20	In transit	GHI012	2024-04-22
5	Jane Smith	456 Elm St	Eve Wilson	654 Pine St	5.00	Pending	KLM045	NULL

-- 12 Retrieve employees whose names contain 'John'

```
SELECT Name , EmployeeID  
FROM employee  
WHERE Name LIKE 'Mark%';
```

The screenshot shows the SQL Editor window with the following code:

```
58  
59      -- 12 Retrieve employees whose names contain 'John'  
60 •  SELECT Name , EmployeeID  
61     FROM employee  
62     WHERE Name LIKE 'Mark%';  
63  
64  
65
```

Below the code is the Result Grid:

	Name	EmployeeID
▶	Mark Johnson	1
*	NULL	NULL

-- 13 Retrieve all courier records with payments greater than \$20

```
SELECT *  
FROM courier INNER JOIN payment ON courier.CourierID=payment.CourierID  
WHERE payment.Amount>=20;
```

The screenshot shows the SQL Editor window with the following code:

```
61     FROM employee  
62     WHERE Name LIKE 'Mark%';  
63  
64      -- 13 Retrieve all courier records with payments greater than $20  
65  
66 •  SELECT *  
67     FROM courier INNER JOIN payment ON courier.CourierID=payment.CourierID  
68     WHERE payment.Amount>=20;  
69  
70  
71
```

Below the code is the Result Grid:

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	PaymentID	CourierID	LocationID	Amount	PaymentDate
2	Alice Johnson	789 Oak St	Bob Brown	321 Maple St	2.00	Delivered	DEP456	2024-04-18	2	2	3	20.00	2024-04-19
3	Eve Wilson	654 Pine St	John Doe	123 Main St	1.80	Pending	GHI789	2024-04-19	3	3	4	25.00	2024-04-20
5	Jane Smith	456 Elm St	Eve Wilson	654 Pine St	5.00	Pending	MNO345	2024-04-20	5	5	1	22.00	2024-04-22

-- 14. Find the total number of couriers handled by each employee

```
ALTER TABLE courier
```

```
ADD COLUMN EmployeeID INT;
```

```
ALTER TABLE courier
```

```
ADD CONSTRAINT EmployeeID FOREIGN KEY (EmployeeID) REFERENCES employee  
(EmployeeID);
```

```
UPDATE Courier SET EmployeeID = 1 WHERE CourierID = 1;
```

```
UPDATE Courier SET EmployeeID = 2 WHERE CourierID = 2;
```

```
UPDATE Courier SET EmployeeID = 1 WHERE CourierID = 3;
```

```
UPDATE Courier SET EmployeeID = 3 WHERE CourierID = 4;
```

```
UPDATE Courier SET EmployeeID = 2 WHERE CourierID = 5;
```

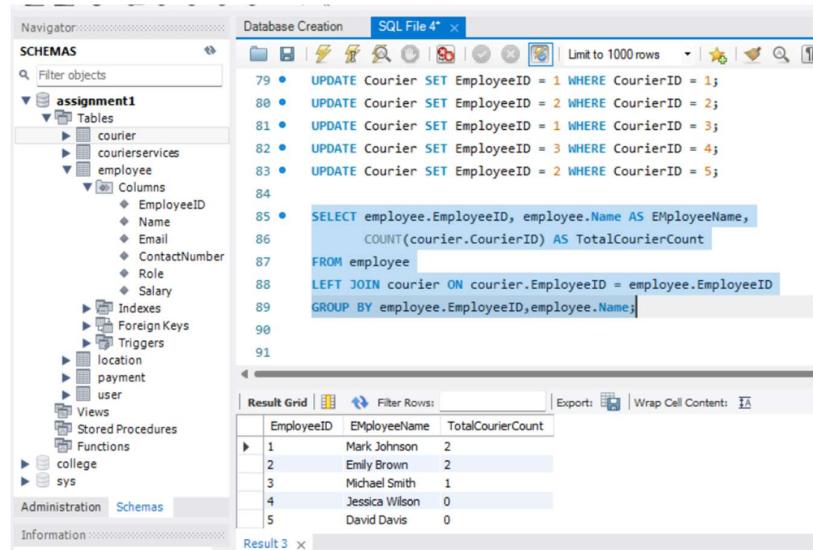
```
SELECT employee.EmployeeID, employee.Name AS EMployeeName,
```

```
    COUNT(courier.CourierID) AS TotalCourierCount
```

```
FROM employee
```

```
LEFT JOIN courier ON courier.EmployeeID = employee.EmployeeID
```

```
GROUP BY employee.EmployeeID,employee.Name;
```



The screenshot shows the SQL Server Management Studio interface. The left pane displays the Navigator with the Schemas node expanded, showing the assignment schema with tables like courier, courierservices, and employee. The right pane shows the Database Creation window with the SQL File 4 tab selected. The SQL pane contains the following code:

```
79 • UPDATE Courier SET EmployeeID = 1 WHERE CourierID = 1;
80 • UPDATE Courier SET EmployeeID = 2 WHERE CourierID = 2;
81 • UPDATE Courier SET EmployeeID = 1 WHERE CourierID = 3;
82 • UPDATE Courier SET EmployeeID = 3 WHERE CourierID = 4;
83 • UPDATE Courier SET EmployeeID = 2 WHERE CourierID = 5;
84
85 • SELECT employee.EmployeeID, employee.Name AS EMployeeName,
86     COUNT(courier.CourierID) AS TotalCourierCount
87 FROM employee
88 LEFT JOIN courier ON courier.EmployeeID = employee.EmployeeID
89 GROUP BY employee.EmployeeID,employee.Name;
```

The Result Grid pane below shows the output of the query:

EmployeeID	EMployeeName	TotalCourierCount
1	Mark Johnson	2
2	Emily Brown	2
3	Michael Smith	1
4	Jessica Wilson	0
5	David Davis	0

-- 15. Calculate the total revenue generated by each location

```
SELECT location.LocationID, location.LocationName ,  
       SUM(payment.Amount) AS TotalRevenue  
  FROM location  
 JOIN payment ON location.LocationID = payment.LocationID  
 GROUP BY location.LocationID,location.LocationName;
```

The screenshot shows the execution of SQL query 15 in a database management tool. The left pane displays the database schema for 'assignment1', including tables such as assignment, courier, courierservices, employee, location, payment, user, and various system tables like sys and information. The right pane shows the query code and its execution results.

Query code:

```
90  
91  
92 -- 15. Calculate the total revenue generated by each location  
93  
94 • SELECT location.LocationID, location.LocationName ,  
95      SUM(payment.Amount) AS TotalRevenue  
96  FROM location  
97  JOIN payment ON location.LocationID = payment.LocationID  
98  GROUP BY location.LocationID,location.LocationName;  
99  
100  
101
```

Execution results:

LocationID	LocationName	TotalRevenue
1	Office 1	22.00
2	Office 2	15.00
3	Office 3	20.00
4	Office 4	25.00
5	Office 5	18.00

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

```
SELECT ReceiverName, ReceiverAddress AS Address,  
       COUNT(*) AS TotalCouriersDelivered  
  FROM Courier  
 GROUP BY ReceiverAddress,ReceiverName;
```

Navigator Database Creation SQL File 4\* | Limit to 1000 rows | Filter Rows: | Export: | Wrap Cell Content: |

```

94 •   SELECT location.LocationID, location.LocationName ,
95           SUM(payment.Amount) AS TotalRevenue
96   FROM location
97   JOIN payment ON location.LocationID = payment.LocationID
98   GROUP BY location.LocationID,location.LocationName;
99
100  -- 16. Find the total number of couriers delivered to each location.
101
102 •   SELECT ReceiverName, ReceiverAddress AS Address,
103           COUNT(*) AS TotalCouriersDelivered
104   FROM Courier
105   GROUP BY ReceiverAddress,ReceiverName;
106

```

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

ReceiverName	Address	TotalCouriersDelivered
Jane Smith	456 Elm St	1
Bob Brown	321 Maple St	1
John Doe	123 Main St	1
Alice Johnson	789 Oak St	1
Eve Wilson	654 Pine St	1

Result 8

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

```

SELECT c.CourierID, AVG(DATEDIFF(c.DeliveryDate,p.PaymentDate)) AS AvgDeliveryTime
FROM courier c
JOIN payment p ON c.CourierID=p.CourierID
group by CourierID
ORDER BY AvgDeliveryTime DESC;

```

Navigator Database Creation SQL File 4\* | Limit to 1000 rows | Filter Rows: | Export: | Wrap Cell Content: |

```

105   GROUP BY ReceiverAddress,ReceiverName;
106
107  -- 17. Find the courier with the highest average delivery time:
108
109 •   SELECT c.CourierID, AVG(DATEDIFF(c.DeliveryDate,p.PaymentDate)) AS AvgDeliveryTime
110   FROM courier c
111   JOIN payment p ON c.CourierID=p.CourierID
112   group by CourierID
113   ORDER BY AvgDeliveryTime DESC;
114
115
116
117

```

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

CourierID	AvgDeliveryTime
1	2.0000
4	1.0000
2	-1.0000
3	HULL
5	HULL

Result 13

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

```
SELECT l.LocationID,l.LocationName,l.Address,p.Amount  
FROM location l  
JOIN payment p ON l.LocationID = p.LocationID  
WHERE p.Amount < 20;
```

The screenshot shows a database interface with a tree view on the left containing tables like courier, courierservices, employee, location, payment, user, and others. The central area displays the SQL query for finding locations with total payments less than 20. The result grid shows two rows of data:

LocationID	LocationName	Address	Amount
2	Office 2	456 Elm St	15.00
5	Office 5	654 Pine St	18.00

## -- 19. Calculate Total Payments per Location

```
SELECT l.LocationID,l.LocationName,SUM(p.Amount) AS TotalPayments  
FROM location l  
JOIN payment p ON l.LocationID = p.LocationID  
GROUP BY l.LocationName, l.LocationID;
```

The screenshot shows a database interface with a tree view on the left containing tables like courier, courierservices, employee, location, payment, user, and others. The central area displays the SQL query for calculating total payments per location. The result grid shows five rows of data:

LocationID	LocationName	TotalPayments
1	Office 1	22.00
2	Office 2	15.00
3	Office 3	20.00
4	Office 4	25.00
5	Office 5	18.00

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

```
SELECT c.*, p.Amount  
FROM courier c  
JOIN payment p ON c.CourierID = p.CourierID  
WHERE p.LocationID = 2  
GROUP BY c.CourierID, p.Amount  
HAVING SUM(p.Amount) > 10;
```

```
127     GROUP BY l.LocationName, l.LocationID;  
128  
129 -- 20. Retrieve couriers who have received payments totaling more than $1000 in a specific location (LocationID = X):  
130  
131 •  SELECT c.CourierID, p.Amount  
132   FROM courier c  
133   JOIN payment p ON c.CourierID = p.CourierID  
134   WHERE p.LocationID = 5 AND p.Amount >=10  
135   group by c.CourierID,p.Amount;  
136  
137  
138  
139
```

CourierID	Amount
4	18.00

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

```
SELECT c.*, p.Amount  
FROM courier c  
JOIN payment p ON c.CourierID = p.CourierID  
WHERE p.PaymentDate > '2024-04-19'  
GROUP BY c.CourierID, p.Amount  
HAVING SUM(p.Amount) > 10;
```

The screenshot shows a database interface with a Navigator pane on the left and a main workspace on the right.

**Navigator:**

- SCHEMAS:**
  - assignment1
  - college
  - sys
- Tables:**
  - courier
  - courierservices
  - employee
  - payment
  - location
  - user
- Views:**
- Stored Procedures:**
- Functions:**

**Main Workspace (SQL File 4\*):**

```

136     HAVING SUM(p.Amount) > 10;
137
138 -- 21. Retrieve couriers who have received payments totaling more than $1000 after a certain date (PaymentDate > 'YYYY-MM-DD'):
139
140 •   SELECT c.*, p.Amount
141   FROM courier c
142   JOIN payment p ON c.CourierID = p.CourierID
143   WHERE p.PaymentDate = '2024-04-19'
144   GROUP BY c.CourierID, p.Amount
145   HAVING SUM(p.Amount) > 10;
146
147
148

```

**Result Grid:**

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	EmployeeID	Amount
2	Alice Johnson	789 Oak St	Bob Brown	321 Maple St	2.00	Delivered	DEF456	2024-04-18	2	20.00

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

```

SELECT l.LocationName, SUM(p.Amount) AS TotalAmountReceived
FROM location l
JOIN payment p ON l.LocationID = p.LocationID
WHERE p.PaymentDate < '2024-04-23'
GROUP BY l.LocationName
HAVING SUM(p.Amount) > 15;

```

The screenshot shows a database interface with a Navigator pane on the left and a main workspace on the right.

**Navigator:**

- SCHEMAS:**
  - assignment1
  - college
  - sys
- Tables:**
  - courier
  - courierservices
  - employee
  - payment
  - location
  - user
- Views:**
- Stored Procedures:**
- Functions:**

**Main Workspace (SQL File 4\*):**

```

147
148 -- 22. Retrieve locations where the total amount received is more than $5000 before a certain date (PaymentDate > 'YYYY-MM-DD')
149
150 •   SELECT l.LocationName, SUM(p.Amount) AS TotalAmountReceived
151   FROM location l
152   JOIN payment p ON l.LocationID = p.LocationID
153   WHERE p.PaymentDate < '2024-04-23'
154   GROUP BY l.LocationName
155   HAVING SUM(p.Amount) > 15;
156
157
158

```

**Result Grid:**

LocationName	TotalAmountReceived
Office 2	33.00
Office 3	42.00
Office 1	25.00

### --- 23. Retrieve Payments with Courier Information

```
SELECT *
FROM courier
JOIN payment ON courier.CourierID = payment.CourierID;
```

The screenshot shows a database interface with a sidebar containing tables like courier, payment, and location. The main area displays a SQL query and its results.

```
-- 23. Retrieve Payments with Courier Information
SELECT *
FROM courier c
JOIN payment p ON c.CourierID = p.CourierID;
```

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	EmployeeID	PaymentID	CourierID	LocationID	Amount	PaymentDate
1	John Doe	123 Main St	Jane Smith	456 Elm St	3.50	In transit	ABC123	2024-04-18	1	1	1	2	15.00	2024-04-18
2	Alice Johnson	789 Oak St	Bob Brown	321 Maple St	2.00	Delivered	DEF456	2024-04-18	2	2	2	3	20.00	2024-04-19
3	Eve Wilson	654 Pine St	John Doe	123 Main St	1.80	Pending	GHI789	2024-04-19	1	3	3	1	25.00	2024-04-20
4	Bob Brown	321 Maple St	Alice Johnson	789 Oak St	4.20	In transit	JKL012	2024-04-22	3	4	4	2	18.00	2024-04-21
5	Jane Smith	456 Elm St	Eve Wilson	654 Pine St	5.00	Pending	MNO345	2024-04-22	2	5	5	3	22.00	2024-04-22

### -- 24. Retrieve Payments with Location Information

```
SELECT *
FROM payment
JOIN location ON location.LocationID = payment.LocationID;
```

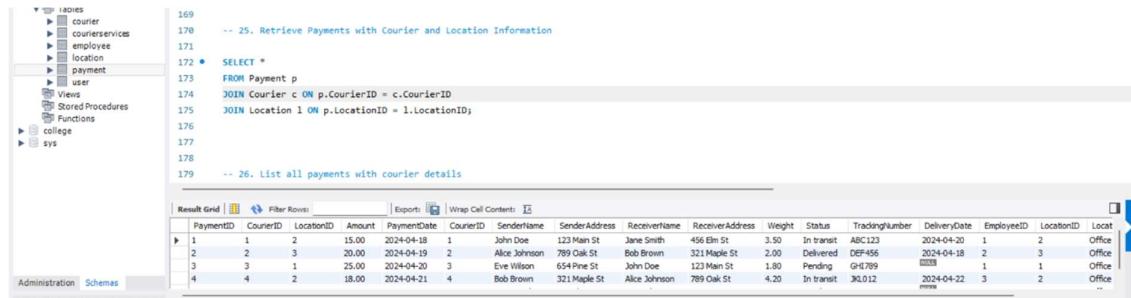
The screenshot shows a database interface with a sidebar containing tables like assignment1, payment, and location. The main area displays a SQL query and its results.

```
-- 24. Retrieve Payments with Location Information
SELECT *
FROM payment
JOIN location ON location.LocationID = payment.LocationID;
```

PaymentID	CourierID	LocationID	Amount	PaymentDate	LocationID	LocationName	Address
1	1	2	15.00	2024-04-18	2	Office 2	456 Elm St
2	2	3	20.00	2024-04-19	3	Office 3	789 Oak St
3	3	1	25.00	2024-04-20	1	Office 1	123 Main St
4	4	2	18.00	2024-04-21	2	Office 2	456 Elm St
5	5	3	22.00	2024-04-22	3	Office 3	789 Oak St

## -- 25. Retrieve Payments with Courier and Location Information

```
SELECT *  
FROM Payment p  
JOIN Courier c ON p.CourierID = c.CourierID  
JOIN Location l ON p.LocationID = l.LocationID;
```

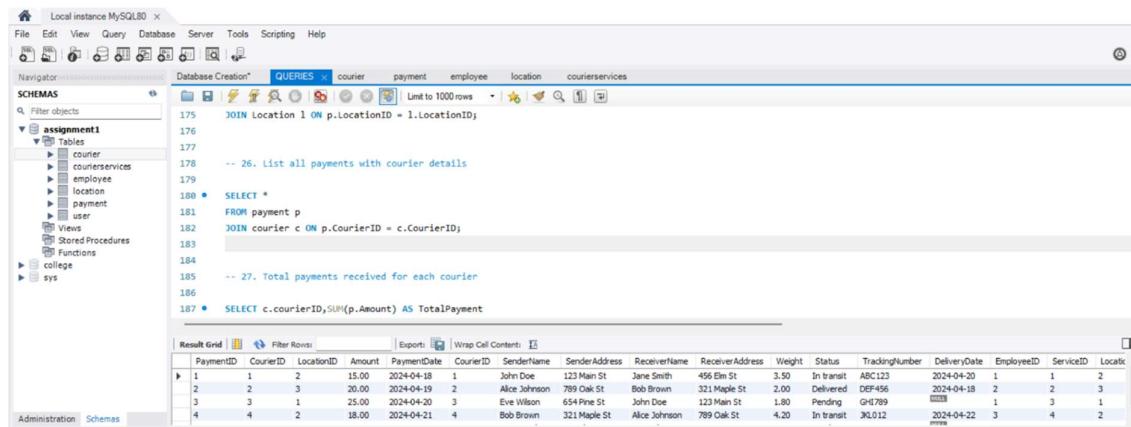


The screenshot shows the MySQL Workbench interface with the query editor containing the code for Exercise 25. The results pane displays a table with 4 rows of payment details, including columns like PaymentID, CourierID, LocationID, Amount, PaymentDate, CourierID, SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate, EmployeeID, LocationID, and Locat. The data is as follows:

PaymentID	CourierID	LocationID	Amount	PaymentDate	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	EmployeeID	LocationID	Locat
1	1	2	15.00	2024-04-18	1	John Doe	123 Main St	Jane Smith	456 Elm St	3.50	In transit	ABC123	2024-04-20	1	2	Office
2	2	3	20.00	2024-04-19	2	Alice Johnson	789 Oak St	Bob Brown	321 Maple St	2.00	Delivered	DEF456	2024-04-18	2	3	Office
3	3	1	25.00	2024-04-20	3	Eve Wilson	654 Pine St	John Doe	123 Main St	1.80	Pending	GHI789	2024-04-22	1	1	Office
4	4	2	18.00	2024-04-21	4	Bob Brown	321 Maple St	Alice Johnson	789 Oak St	4.20	In transit	JKL012	2024-04-22	3	2	Office

## -- 26. List all payments with courier details

```
SELECT *  
FROM payment p  
JOIN courier c ON p.CourierID = c.CourierID;
```



The screenshot shows the MySQL Workbench interface with the query editor containing the code for Exercise 26. The results pane displays a table with 4 rows of payment details, including columns like PaymentID, CourierID, LocationID, Amount, PaymentDate, CourierID, SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate, EmployeeID, ServiceID, and Locat. The data is as follows:

PaymentID	CourierID	LocationID	Amount	PaymentDate	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	EmployeeID	ServiceID	Locat
1	1	2	15.00	2024-04-18	1	John Doe	123 Main St	Jane Smith	456 Elm St	3.50	In transit	ABC123	2024-04-20	1	2	2
2	2	3	20.00	2024-04-19	2	Alice Johnson	789 Oak St	Bob Brown	321 Maple St	2.00	Delivered	DEF456	2024-04-18	2	3	3
3	3	1	25.00	2024-04-20	3	Eve Wilson	654 Pine St	John Doe	123 Main St	1.80	Pending	GHI789	2024-04-22	1	3	1
4	4	2	18.00	2024-04-21	4	Bob Brown	321 Maple St	Alice Johnson	789 Oak St	4.20	In transit	JKL012	2024-04-22	3	4	2

-- 27. Total payments received for each courier

```
SELECT c.courierID, SUM(p.Amount) AS TotalPayment  
FROM courier c  
JOIN payment p ON c.CourierID = p.CourierID  
group by c.CourierID;
```

Navigator Schemas

SCHEMAS

assignment1

Tables

- courier
- courservices
- employee
- location
- payment
- user

Views

Stored Procedures

Functions

college

sys

Database Creation QUERIES x

183  
184  
185 -- 27. Total payments received for each courier  
186  
187 • 187 • SELECT c.courierID, SUM(p.Amount) AS TotalPayment  
188 FROM courier c  
189 JOIN payment p ON c.CourierID = p.CourierID  
190 group by c.CourierID;  
191  
192  
193  
194  
195

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

courierID	TotalPayment
1	15.00
2	20.00
3	25.00
4	18.00
5	22.00

Administration Schemas Information Result 14 x

-- 28. List payments made on a specific date

```
SELECT *  
FROM payment  
WHERE PaymentDate = '2024-04-20';
```

Navigator Schemas

SCHEMAS

assignment1

Tables

- user
- Views
- Stored Procedures
- Functions

college

sys

Database Creation QUERIES x

194  
195 -- 28. List payments made on a specific date  
196  
197 • 197 • SELECT \*  
198 FROM payment  
199 WHERE PaymentDate = '2024-04-20';  
200

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

PaymentID	CourierID	LocationID	Amount	PaymentDate
3	3	1	25.00	2024-04-20
HULL	HULL	HULL	HULL	HULL

Administration Schemas

## -- 29. Get Courier Information for Each Payment

```
SELECT p.PaymentID, c.*  
FROM courier c  
JOIN payment p ON c.CourierID = p.CourierID;
```

The screenshot shows the MySQL Workbench interface. The top bar includes 'File', 'Edit', 'View', 'Query', 'Database', 'Server', 'Tools', 'Scripting', and 'Help'. The 'Navigator' pane on the left lists 'SCHEMAS' (assignment1), 'Tables' (courier, courierservices, employee, location, payment, user), and 'Views', 'Stored Procedures', 'Functions', 'college', and 'sys'. The 'QUERIES' tab in the center contains the following code:

```
195 • SELECT c.CourierID , c.SenderName , c.Status  
196 FROM courier c  
197 JOIN payment p ON c.CourierID = p.CourierID  
198 WHERE PaymentDate = '2024-04-20';  
199  
200 -- 29. Get Courier Information for Each Payment  
201  
202 • SELECT p.PaymentID, c.*  
203 FROM courier c  
204 JOIN payment p ON c.CourierID = p.CourierID;  
205  
206 /*-- 30. Get Payment Details with Location  
207 -- 31. Calculating Total Payments for Each Courier
```

The 'Result Grid' below displays the results of the second query:

PaymentID	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	EmployeeID
1	1	John Doe	123 Main St	Jane Smith	456 Elm St	3.50	In transit	ABC123	2024-04-20	1
2	2	Alice Johnson	789 Oak St	Bob Brown	321 Maple St	2.00	Delivered	DEF456	2024-04-18	2
3	3	Eve Wilson	654 Pine St	John Doe	123 Main St	1.80	Pending	GHI789		1
4	4	Bob Brown	321 Maple St	Alice Johnson	789 Oak St	4.20	In transit	JKL012	2024-04-22	3
5	5	Jane Smith	456 Elm St	Eve Wilson	654 Pine St	5.00	Pending	MNO345		2

## -- 30. Get Payment Details with Location

```
SELECT l.LocationID, p.*  
FROM payment p  
JOIN location l ON p.LocationID = l.LocationID;
```

The screenshot shows the MySQL Workbench interface. The top bar includes 'File', 'Edit', 'View', 'Query', 'Database', 'Server', 'Tools', 'Scripting', and 'Help'. The 'Navigator' pane on the left lists 'SCHEMAS' (assignment1), 'Tables' (courier, courierservices, employee, location, payment, user), and 'Views', 'Stored Procedures', 'Functions', 'college', and 'sys'. The 'QUERIES' tab in the center contains the following code:

```
206 -- 30. Get Payment Details with Location  
207  
208 • SELECT l.LocationID, p.*  
209 FROM payment p  
210 JOIN location l ON p.LocationID = l.LocationID  
211  
212  
213 /*-- 31. Calculating Total Payments for Each Courier
```

The 'Result Grid' below displays the results of the second query:

LocationID	PaymentID	CourierID	LocationID	Amount	PaymentDate
2	1	1	2	15.00	2024-04-18
3	2	2	3	20.00	2024-04-19
1	3	3	1	25.00	2024-04-20
2	4	4	2	18.00	2024-04-21
3	5	5	3	22.00	2024-04-22

### -- 31. Calculating Total Payments for Each Courier

```
SELECT c.CourierID, SUM(p.Amount) AS TotalPayment  
FROM courier c  
JOIN payment p ON c.CourierID = p.CourierID  
group by c.CourierID;
```

The screenshot shows a database interface with a 'QUERIES' tab selected. On the left, a 'Navigator' pane displays the schema structure under 'SCHEMAS'. The 'assignment1' schema is expanded, showing tables like 'courier', 'payment', and 'user'. Below the schema tree are 'Views', 'Stored Procedures', and 'Functions'. Under 'college' and 'sys', there are no tables listed. The main area contains the SQL code for query 208, which calculates total payments for each courier. The result grid shows five rows of data:

CourierID	TotalPayment
1	15.00
2	20.00
3	25.00
4	18.00
5	22.00

Below the result grid, a message says 'Result 19'.

### -- 32. List Payments Within a Date Range

```
SELECT *  
FROM payment  
WHERE PaymentDate BETWEEN '2024-04-19' AND '2024-04-21';
```

Navigator: Database Creation QUERIES\*

Filter objects

**SCHEMAS**

- assignment1
  - Tables: courier, courierservices, employee, location, payment, user
  - Views
  - Stored Procedures
  - Functions
- college
- sys

216     **FROM** courier c  
 217     **JOIN** payment p **ON** c.CourierID = p.CourierID  
 218     **group by** c.CourierID;  
 219  
 220  
 221     -- 32. List Payments Within a Date Range  
 222  
 223 •   **SELECT** \*  
 224     **FROM** payment  
 225     **WHERE** PaymentDate **BETWEEN** '2024-04-19' **AND** '2024-04-21';  
 226  
 227  
 228     /\*--33. Retrieve a list of all users and their corresponding

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

	PaymentID	CourierID	LocationID	Amount	PaymentDate
▶	2	2	3	20.00	2024-04-19
	3	3	1	25.00	2024-04-20
	4	4	2	18.00	2024-04-21
*	NULL	NULL	NULL	NULL	NULL

Administration Schemas Information

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

```
SELECT *
  FROM courier c
  LEFT JOIN user u ON c.SenderName = u.Name
  UNION
SELECT *
  FROM courier c
  RIGHT JOIN user u ON c.SenderName = u.Name;
```

Filter objects

**SCHEMAS**

- assignment1
  - Tables: courier, courierservices, employee, location, payment, user
  - Views
  - Stored Procedures
  - Functions
- college
- sys

229  
 230     -- 33. Retrieve a list of all users and their corresponding courier records, including cases where there are no matches on either side  
 231  
 232 •   **SELECT** \*  
 233     **FROM** courier c  
 234     **LEFT JOIN** user u **ON** c.SenderName = u.Name  
 235     UNION  
 236     **SELECT** \*  
 237     **FROM** courier c  
 238     **RIGHT JOIN** user u **ON** c.SenderName = u.Name;  
 239  
 240  
 241

Result Grid | Filter Rows: Export: Wrap Cell Content: Result 25 x

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	EmployeeID	ServiceID	UserID	Name	Email	Password	ContactNo
1	John Doe	123 Main St	Jane Smith	456 Elm St	3.50	In transit	ABC123	2024-04-20	1	1	1	John Doe	john@example.com	password123	12345678
2	Alice Johnson	789 Oak St	Bob Brown	321 Maple St	2.00	Delivered	DEF456	2024-04-18	2	2	3	Alice Johnson	alice@example.com	password123	98765432
3	Eve Wilson	654 Pine St	John Doe	123 Main St	1.80	Pending	GHI789	2024-04-21	1	3	5	Eve Wilson	eve@example.com	passworddefg	78945612
4	Bob Brown	321 Maple St	Alice Johnson	789 Oak St	4.20	In transit	JKL012	2024-04-22	3	4	4	Bob Brown	bob@example.com	passwordabc	45612378

Administration Schemas Information

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

```
SELECT *
FROM courier c
LEFT JOIN courierservices s ON c.ServiceID = s.ServiceID
UNION
SELECT *
FROM courier c
RIGHT JOIN courierservices s ON c.ServiceID = s.ServiceID;
```

The screenshot shows the execution of the SQL query in a database management tool. The code is as follows:

```
241
242
243 -- 34. Retrieve a list of all couriers and their corresponding services, including cases where there are no matches on either side
244
245 • SELECT *
246 FROM courier c
247 LEFT JOIN courierservices s ON c.ServiceID = s.ServiceID
248 UNION
249 SELECT *
250 FROM courier c
251 RIGHT JOIN courierservices s ON c.ServiceID = s.ServiceID;
252
253
```

The results grid displays 26 rows of data:

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	EmployeeID	ServiceID	ServiceID	ServiceName	Cost
1	John Doe	123 Main St	Jane Smith	456 Elm St	3.50	In transit	ABC123	2024-04-20	1	1	1	Standard	10.00
2	Alice Johnson	789 Oak St	Bob Brown	321 Maple St	2.00	Delivered	DEF456	2024-04-18	2	2	2	Express	20.00
3	Eve Wilson	654 Pine St	John Doe	123 Main St	1.80	Pending	GHI789	2024-04-21	1	3	3	Same Day	30.00
4	Bob Brown	321 Maple St	Alice Johnson	789 Oak St	4.20	In transit	JKL012	2024-04-22	3	4	4	Oversight	15.00
5	Jane Smith	456 Elm St	Eve Wilson	654 Pine St	5.00	Pending	MNO345	2024-04-23	2	5	5	International	50.00

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

```
SELECT e.*, p.*
FROM employee e
LEFT JOIN courier c ON c.EmployeeID = e.EmployeeID
LEFT JOIN payment p ON c.CourierID = p.CourierID
UNION
SELECT e.*, p.*
FROM employee e
RIGHT JOIN courier c ON c.EmployeeID = e.EmployeeID
RIGHT JOIN payment p ON c.CourierID = p.CourierID;
```

```

256
257      -- 35. Retrieve a list of all employees and their corresponding payments, including cases where there are no matches on either side
258
259 •   SELECT e.*, p.*
260     FROM employee e
261     LEFT JOIN courier c ON c.EmployeeID = e.EmployeeID
262     LEFT JOIN payment p ON c.CourierID = p.CourierID
263
264 UNION
265
266     SELECT e.*, p.*
267     FROM employee e
268     RIGHT JOIN courier c ON c.EmployeeID = e.EmployeeID
269     RIGHT JOIN payment p ON c.CourierID = p.CourierID;
270

```

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

EmployeeID	Name	Email	ContactNumber	Role	Salary	PaymentID	CourierID	LocationID	Amount	PaymentDate
1	Mark Johnson	mark@example.com	1239876540	Manager	50000.00	1	1	2	15.00	2024-04-18
1	Mark Johnson	mark@example.com	1239876540	Manager	50000.00	3	3	1	25.00	2024-04-20
2	Emily Brown	emily@example.com	9876543210	Clerk	30000.00	2	2	3	20.00	2024-04-19
2	Emily Brown	emily@example.com	9876543210	Clerk	30000.00	5	5	3	22.00	2024-04-22
3	Michael Smith	michael@example.com	4567890123	Driver	35000.00	4	4	2	18.00	2024-04-21

Result 27 ×

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

```

SELECT *
FROM courier c
CROSS JOIN courierservices s ON c.ServiceID = s.ServiceID; SELECT *
FROM courier c
CROSS JOIN courierservices s ON c.ServiceID = s.ServiceID;

```

```

272
273      -- 36. List all users and all courier services, showing all possible combinations.
274
275 •   SELECT *
276     FROM courier c
277     CROSS JOIN courierservices s ON c.ServiceID = s.ServiceID;
278
279
280
281
282
283

```

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

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	EmployeeID	ServiceID	ServiceID	ServiceName	Cost
1	John Doe	123 Main St	Jane Smith	456 Elm St	3.50	In transit	ABC123	2024-04-20	1	1	1	Standard	10.00
2	Alice Johnson	789 Oak St	Bob Brown	321 Maple St	2.00	Delivered	DEF456	2024-04-18	2	2	2	Express	20.00
3	Eve Wilson	654 Pine St	John Doe	123 Main St	1.80	Pending	GHI789	NULL	1	3	3	Same Day	30.00
4	Bob Brown	321 Maple St	Alice Johnson	789 Oak St	4.20	In transit	JKL012	2024-04-22	3	4	4	Overnight	15.00
5	Jane Smith	456 Elm St	Eve Wilson	654 Pine St	5.00	Pending	MNO345	NULL	2	5	5	International	50.00

Result 28 ×

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

```
SELECT e.* , l.*  
FROM location l  
CROSS JOIN payment p ON l.LocationID = p.LocationID  
CROSS JOIN courier c ON p.CourierID = c.CourierID  
CROSS JOIN employee e ON c.EmployeeID = e.EmployeeID;
```

```
284  
285      -- 37. List all employees and all locations, showing all possible combinations:  
286  
287 •   SELECT e.* , l.*  
288     FROM location l  
289     CROSS JOIN payment p ON l.LocationID = p.LocationID  
290     CROSS JOIN courier c ON p.CourierID = c.CourierID  
291     CROSS JOIN employee e ON c.EmployeeID = e.EmployeeID;  
292  
293  
294  
295
```

	CourierID	UserID	Name	Email	Password	ContactNumber	Address	Status	TrackingNumber	DeliveryDate
▶	1	1	John Doe	john@example.com	password123	123 Main St	In transit	ABC123	2024-04-20	
	2	3	Alice Johnson	alice@example.com	password789	9876543210	789 Oak St	Delivered	DEF456	2024-04-18
	3	5	Eve Wilson	eve@example.com	passworddefg	7894561230	654 Pine St	Pending	GHI789	NULL
	4	4	Bob Brown	bob@example.com	passwordabc	4561237890	321 Maple St	In transit	JKL012	2024-04-22
	5	2	Jane Smith	jane@example.com	password456	0987654321	456 Elm St	Pending	MNO345	NULL

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

```
SELECT c.CourierID, u.*, c.Status, c.TrackingNumber, c.DeliveryDate
```

```
FROM courier c
```

```
LEFT JOIN user u ON c.SenderName = u.Name;
```

```
297  
298      -- 38. Retrieve a list of couriers and their corresponding sender information (if available)  
299  
300 •   SELECT c.CourierID, u.*, c.Status, c.TrackingNumber, c.DeliveryDate  
301     FROM courier c  
302     LEFT JOIN user u ON c.SenderName = u.Name;  
303  
304  
305  
306  
307
```

	CourierID	UserID	Name	Email	Password	ContactNumber	Address	Status	TrackingNumber	DeliveryDate
▶	1	1	John Doe	john@example.com	password123	123 Main St	In transit	ABC123	2024-04-20	
	2	3	Alice Johnson	alice@example.com	password789	9876543210	789 Oak St	Delivered	DEF456	2024-04-18
	3	5	Eve Wilson	eve@example.com	passworddefg	7894561230	654 Pine St	Pending	GHI789	NULL
	4	4	Bob Brown	bob@example.com	passwordabc	4561237890	321 Maple St	In transit	JKL012	2024-04-22
	5	2	Jane Smith	jane@example.com	password456	0987654321	456 Elm St	Pending	MNO345	NULL

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

```
SELECT CourierID, ReceiverName, ReceiverAddress , Status  
FROM courier ;
```

The screenshot shows the SQL Server Management Studio interface. On the left, there's a tree view of the database schema under 'assignment1'. The 'Tables' node is expanded, showing 'courier', 'courierservices', 'employee', 'location', 'payment', and 'user'. Below these are 'Views', 'Stored Procedures', and 'Functions'. Further down are 'college' and 'sys' nodes. The main pane displays the following code:

```
285  
286 -- 39. Retrieve a list of couriers and their corresponding receiver information (if available):  
287  
288 • SELECT CourierID, ReceiverName, ReceiverAddress , Status  
289 FROM courier ;  
290  
291  
292  
293  
294  
295  
296 /*40. Retrieve a list of couriers along with the courier service details (if available):
```

Below the code, a 'Result Grid' shows the query results:

CourierID	ReceiverName	ReceiverAddress	Status
1	Jane Smith	456 Elm St	In transit
2	Bob Brown	321 Maple St	Delivered
3	John Doe	123 Main St	Pending
4	Alice Johnson	789 Oak St	In transit
5	Eve Wilson	654 Pine St	Pending

The status bar at the bottom of the grid says 'courier 46'.

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

```
SELECT c.CourierID,s.*  
FROM courier c  
LEFT JOIN courierservices s ON c.ServiceID = s.ServiceID;
```

The screenshot shows the SQL Server Management Studio interface. On the left, there's a tree view of the database schema under 'assignment1'. The 'Tables' node is expanded, showing 'courier', 'courierservices', 'employee', 'location', 'payment', and 'user'. Below these are 'Views', 'Stored Procedures', and 'Functions'. Further down are 'college' and 'sys' nodes. The main pane displays the following code:

```
293  
294  
295 -- 40. Retrieve a list of couriers along with the courier service details (if available):  
296  
297 • SELECT c.CourierID,s.*  
298 FROM courier c  
299 LEFT JOIN courierservices s ON c.ServiceID = s.ServiceID  
300  
301  
302  
303  
304
```

Below the code, a 'Result Grid' shows the query results:

CourierID	ServiceID	ServiceName	Cost
1	1	Standard	10.00
2	2	Express	20.00
3	3	Same Day	30.00
4	4	Overnight	15.00
5	5	International	50.00

The status bar at the bottom of the grid says 'Result 48'.

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

```
SELECT e.EmployeeID, e.Name AS EmployeeName, COUNT(c.CourierID) AS  
NumCouriersAssigned  
FROM Employee e  
LEFT JOIN courier c ON e.EmployeeID = c.EmployeeID  
GROUP BY e.EmployeeID, e.Name;
```

The screenshot shows a database interface with a sidebar containing a tree view of database objects under the schema 'assignment1'. The 'Tables' node is expanded, showing 'courier', 'courierservices', 'employee', 'location', 'payment', and 'user'. Other nodes like 'Views', 'Stored Procedures', and 'Functions' are also visible. Below the sidebar, the main area displays the SQL query for question 41. The query is highlighted in blue. The results are shown in a grid table titled 'Result Grid'.

	EmployeeID	EmployeeName	NumCouriersAssigned
▶	1	Mark Johnson	2
	2	Emily Brown	2
	3	Michael Smith	1
	4	Jessica Wilson	0
	5	David Davis	0

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

```
SELECT l.LocationID , SUM(p.Amount) AS TotalPaymentAmout  
FROM location l  
LEFT JOIN payment p ON l.LocationID = p.LocationID  
GROUP BY l.LocationID;
```

▶ courier  
▶ courierservices  
▶ employee  
▶ location  
▶ payment  
▶ user

Views  
Stored Procedures  
Functions

▶ college  
▶ sys

```

317 -- 42. Retrieve a list of locations and the total payment amount received at each location:
318
319 • SELECT l.LocationID , SUM(p.Amount) AS TotalPaymentAmount
320 FROM location l
321 LEFT JOIN payment p ON l.LocationID = p.LocationID
322 GROUP BY l.LocationID
323
324
325
326 /*43. Retrieve all couriers sent by the same sender (based on SenderName).*/

```

LocationID	TotalPaymentAmount
1	25.00
2	33.00
3	42.00
4	NULL
5	NULL

Administration Schemas

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

```

SELECT SenderName, CourierID
FROM courier
GROUP BY SenderName,CourierID;

```

▶ courierservices  
▶ employee  
▶ location  
▶ payment  
▶ user

Views  
Stored Procedures  
Functions

▶ college  
▶ sys

```

326 -- 43. Retrieve all couriers sent by the same sender (based on SenderName).
327
328 • SELECT SenderName, CourierID
329 FROM courier
330 GROUP BY SenderName,CourierID;
331
332
333
334 /*44. List all employees who share the same role.

```

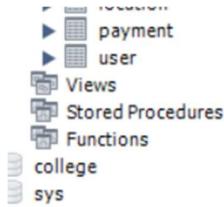
SenderName	CourierID
John Doe	1
Alice Johnson	2
Eve Wilson	3
Bob Brown	4
Jane Smith	5

Administration Schemas

Information

-- 44. List all employees who share the same role.

```
SELECT Role, GROUP_CONCAT(Name) AS Employees  
FROM Employee  
GROUP BY Role;
```

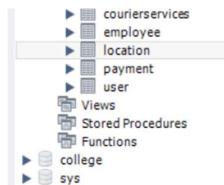


```
332  
333      -- 44. List all employees who share the same role.  
334  
335 •   SELECT Role, GROUP_CONCAT(Name) AS Employees  
336     FROM Employee  
337     GROUP BY Role;  
338  
339
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	Role	Employees		
▶	Administrator	Jessica Wilson		
	Clerk	Emily Brown		
	Courier	David Davis		
	Driver	Michael Smith		
	Manager	Mark Johnson		

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

```
SELECT p.*, l.LocationName  
FROM Payment p  
JOIN courier c ON p.CourierID = c.CourierID  
JOIN location l ON c.SenderAddress = l.Address;
```



```
341  
342      -- 45. Retrieve all payments made for couriers sent from the same location.  
343  
344 •   SELECT p.*, l.LocationName  
345     FROM Payment p  
346     JOIN courier c ON p.CourierID = c.CourierID  
347     JOIN location l ON c.SenderAddress = l.Address;  
348  
349  
350      -- 46. Retrieve all couriers sent from the same location (based on SenderAddress).
```

Result Grid					
PaymentID	CourierID	LocationID	Amount	PaymentDate	LocationName
1	1	2	15.00	2024-04-18	Office 1
5	5	3	22.00	2024-04-22	Office 2
2	2	3	20.00	2024-04-19	Office 3
4	4	2	18.00	2024-04-21	Office 4
3	3	1	25.00	2024-04-20	Office 5

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

```
SELECT SenderName, CourierID, SenderAddress  
FROM courier  
GROUP BY SenderName,CourierID,SenderAddress ;
```

The screenshot shows a database interface with a sidebar containing icons for payment, user, views, stored procedures, functions, college, and sys. The main area displays a SQL query and its results.

```
349  
350 -- 46. Retrieve all couriers sent from the same location (based on SenderAddress).  
351  
352 • SELECT SenderName, CourierID, SenderAddress  
353 FROM courier  
354 GROUP BY SenderName,CourierID,SenderAddress ;  
355  
356
```

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

SenderName	CourierID	SenderAddress
John Doe	1	123 Main St
Alice Johnson	2	789 Oak St
Eve Wilson	3	654 Pine St
Bob Brown	4	321 Maple St
Jane Smith	5	456 Elm St

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

```
SELECT e.Name, COUNT(c.CourierID) AS TotalNumOfCouriers  
FROM employee e  
LEFT JOIN courier c ON e.EmployeeID = c.EmployeeID  
GROUP BY e.Name ;
```

The screenshot shows a database interface with a sidebar containing icons for views, stored procedures, functions, college, and sys. The main area displays a SQL query and its results.

```
358  
359 -- 47. List employees and the number of couriers they have delivered:  
360  
361 • SELECT e.Name, COUNT(c.CourierID) AS TotalNumOfCouriers  
362 FROM employee e  
363 LEFT JOIN courier c ON e.EmployeeID = c.EmployeeID
```

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

Name	TotalNumOfCouriers
Mark Johnson	2
Emily Brown	2
Michael Smith	1
Jessica Wilson	0
David Davis	0

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

```
SELECT c.CourierID,s.ServiceName,s.Cost AS ServiceCost,p.Amount AS PaidAmount  
FROM payment p  
LEFT JOIN courier c ON p.CourierID = c.CourierID  
LEFT JOIN courierservices s ON c.ServiceID = s.ServiceID  
WHERE p.Amount > s.Cost;
```

```
    ► counter
    ► courierservices
    ► employee
    ► location
    ► payment
    ► user
    ► Views
    ► Stored Procedures
    ► Functions
    ► college
    ► sys

367
368    -- 48. Find couriers that were paid an amount greater than the cost of their respective courier services *
369
370 • SELECT c.CourierID,s.ServiceName,s.Cost AS ServiceCost,p.Amount AS PaidAmount
371     FROM payment p
372     LEFT JOIN courier c ON p.CourierID = c.CourierID
373     LEFT JOIN courierservices s ON c.ServiceID = s.ServiceID
374     WHERE p.Amount > s.Cost
375
376
```

---

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

	CourierID	ServiceName	ServiceCost	PaidAmount
►	1	Standard	10.00	15.00
►	4	Oversight	15.00	18.00

-- 49. Find couriers that have a weight greater than the average weight of all couriers

```
SELECT *\nFROM courier\nWHERE Weight > (SELECT AVG(Weight) FROM courier);
```

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

```
SELECT *  
FROM employee  
WHERE Salary > (SELECT AVG(Salary) FROM employee);
```

```
383      -- 50. Find the names of all employees who have a salary greater than the average salary:  
384  
385 •  SELECT *  
386   FROM employee  
387   WHERE Salary > (SELECT AVG(Salary) FROM employee);  
388  
389  
390  
391  
392  
393
```

Result Grid						
	EmployeeID	Name	Email	ContactNumber	Role	Salary
▶	1	Mark Johnson	mark@example.com	1239876540	Manager	50000.00
▶	4	Jessica Wilson	jessica@example.com	3216549870	Administrator	40000.00
*	HULL	HULL	HULL	HULL	HULL	HULL

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

```
SELECT *  
FROM courierservices  
WHERE Cost < (SELECT MAX(Cost) FROM courierservices );
```

```
391      -- 51. Find the total cost of all courier services where the cost is less than the maximum cost  
392  
393 •  SELECT *  
394   FROM courierservices  
395   WHERE Cost < (SELECT MAX(Cost) FROM courierservices );  
396  
397
```

Result Grid			
	ServiceID	ServiceName	Cost
▶	1	Standard	10.00
▶	2	Express	20.00
▶	3	Same Day	30.00
▶	4	Overnight	15.00
*	HULL	HULL	HULL

-- 52. Find all couriers that have been paid for

```
SELECT *  
FROM courier c  
INNER JOIN payment p ON c.CourierID = p.CourierID;
```

The screenshot shows a database interface with a sidebar containing schema and object lists. The main area displays a query result grid titled 'Result Grid' with 74 rows. The columns include CourierID, SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate, EmployeeID, ServiceID, PaymentID, CourierID, LocationID, Amount, and PaymentDate. The data shows various transactions between couriers and locations.

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	EmployeeID	ServiceID	PaymentID	CourierID	LocationID	Amount	PaymentDate
1	John Doe	123 Main St	Jane Smith	456 Elm St	3.50	In transit	ABC123	2024-04-20	1	1	1	1	2	15.00	2024-04-18
2	Alice Johnson	789 Oak St	Bob Brown	321 Maple St	2.00	Delivered	DEF456	2024-04-18	2	2	2	2	3	20.00	2024-04-19
3	Eve Wilson	654 Pine St	John Doe	123 Main St	1.80	Pending	GHI789	2024-04-19	1	3	3	3	1	25.00	2024-04-20
4	Bob Brown	321 Maple St	Alice Johnson	789 Oak St	4.20	In transit	JKL012	2024-04-22	3	4	4	4	2	18.00	2024-04-21
5	Jane Smith	456 Elm St	Eve Wilson	654 Pine St	5.00	Pending	MNO345	2024-04-22	2	5	5	5	3	22.00	2024-04-22

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

```
SELECT l.*, p.Amount  
FROM location l  
INNER JOIN payment p ON l.LocationID = p.LocationID  
ORDER BY p.Amount DESC;
```

The screenshot shows a database interface with a sidebar containing schema and object lists. The main area displays a query result grid titled 'Result Grid' with 5 rows. The columns include LocationID, LocationName, Address, and Amount. The data shows five locations with their respective addresses and total payment amounts.

LocationID	LocationName	Address	Amount
1	Office 1	123 Main St	25.00
3	Office 3	789 Oak St	22.00
3	Office 3	789 Oak St	20.00
2	Office 2	456 Elm St	18.00
2	Office 2	456 Elm St	15.00

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

```
SELECT *  
FROM courier  
WHERE Weight > ( SELECT SUM(Weight)  
FROM courier  
WHERE SenderName = 'Eve Wilson')
```