Assignments-1

A database administrator for a fictional company named "TechShop," which sells electronic gadgets. TechShop maintains data related to their products, customers, and orders. Task is to design and implement a database for TechShop based on the following requirements.

TASK - 1 DATABASE DESIGN

1) Create a database named “TechShop” .

```
ysql> create DATABASE TECHSHOP;
uery OK, 1 row affected (0.03 sec)
```

2) Create the schema for customers,products,OrderDetails,Inventory tables based on the provided schema.

Customers

```
mysql>  CREATE TABLE Customers(
    ->  CustomerID INTEGER PRIMARY KEY,
    ->  FirstName VARCHAR(15),
    ->  LastName VARCHAR(15),
    -> Email VARCHAR(30),
    -> Phone INTEGER,
    -> Address  VARCHAR(50)
    -> );
```

Products

```
mysql> CREATE TABLE Products(
    ->  productID INTEGER PRIMARY KEY,
    -> productName VARCHAR(15),
    -> Description VARCHAR(50),
    -> Price INTEGER
    -> );
```

Orders

```
mysql> CREATE TABLE Orders(
    -> OrderID INTEGER PRIMARY KEY,
    -> CustomerID INTEGER,
    -> OrderDate DATE,
    -> TotalAmount INTEGER,
    -> FOREIGN KEY(CustomerID) REFERENCES Customers(CustomerID)
    -> );
```

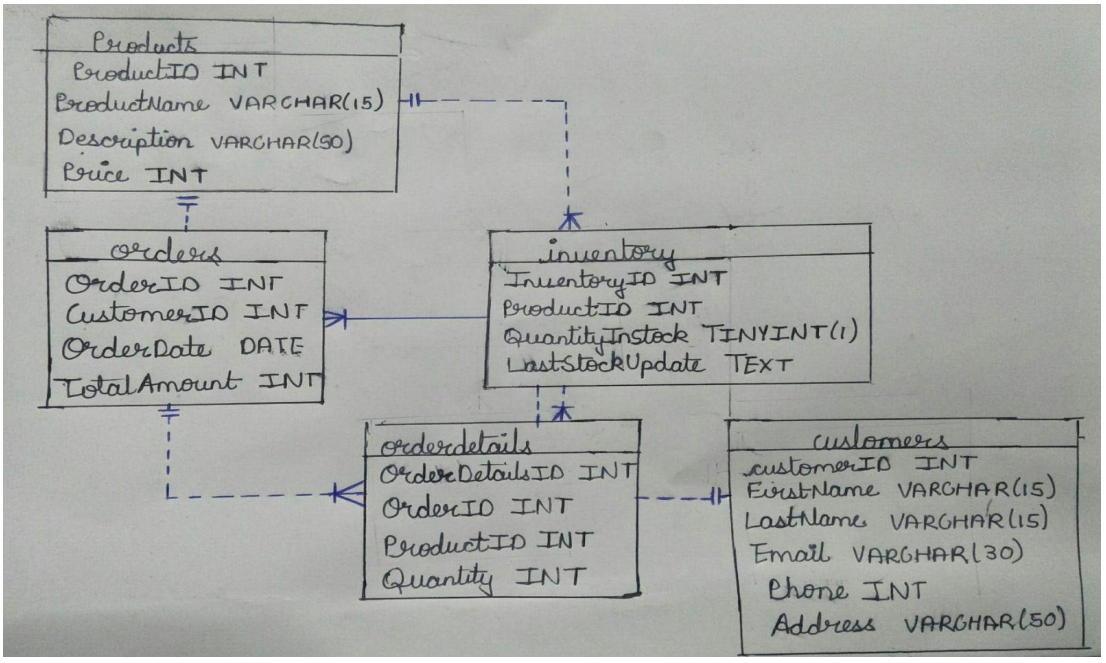
OrderDetails

```
mysql> CREATE TABLE OrderDetails(
    ->  OrderDetailsID INTEGER PRIMARY KEY,
    -> OrderID INTEGER ,
    -> ProductID INTEGER,
    -> Quantity INTEGER,
    -> FOREIGN KEY(OrderID) REFERENCES Orders(OrderID),
    -> FOREIGN KEY(ProductID) REFERENCES Products(ProductID)
    -> );
```

Inventory

```
mysql> CREATE TABLE Inventory(
    ->  InventoryID INTEGER PRIMARY KEY,
    ->  ProductID INTEGER ,
    -> QuantityInStock BOOLEAN,
    -> LastStockUpdate text,
    -> FOREIGN KEY(ProductID) references Products(productID)
    -> );
```

3) Create E-R(entity relationship diagram) for the database.



4) Create and appropriate primary key and foreign key constraints for referential integrity

Primary key

TABLE_NAME	CONSTRAINT_NAME
customers	PRIMARY
products	PRIMARY
orders	PRIMARY
orderdetails	PRIMARY
inventory	PRIMARY

Foreign key

TABLE_NAME	CONSTRAINT_NAME
orders	orders_ibfk_1
orderdetails	orderdetails_ibfk_1
orderdetails	orderdetails_ibfk_2
inventory	inventory_ibfk_1

5) Insert at least 10 sample record into each of the following tables .

Customers

CustomerID	FirstName	LastName	Email	Phone	Address
1	John	Doe	john@example.com	1234567890	123 Main St
2	Alice	Smith	alice@example.com	1876543210	456 Elm St
3	Bob	Johnson	bob@example.com	1551234567	789 Oak St
4	Emily	Brown	emily@example.com	1779998888	101 Pine St
5	Michael	Davis	michael@example.com	1445556666	246 Maple St
6	Sarah	Wilson	sarah@example.com	1223334444	369 Cedar St
7	David	Miller	david@example.com	1887776666	482 Birch St
8	Olivia	Garcia	olivia@example.com	1661112222	573 Spruce St
9	Sophia	Martinez	sophia@example.com	1334445555	798 Walnut St
10	James	Thompson	james@example.com	1990001111	854 Cherry St

Products

ProductID	ProductName	Description	Price
200	Smart TV	smart features, and seamless connectivity. Stream your favorite content, and experience a new level of home entertainment.	79900
202	Washing Machine	Simplify laundry days with our advanced washing machine. Choose from various settings, enjoy energy and let its smart technology take care of your clothes.	6990
203	Refrigerator	Keep your food fresh and organized with our modern refrigerator. Innovative features, ample storage, and efficient cooling make it a perfect addition to your kitchen.	8990
204	Microwave Oven	Effortlessly prepare meals with our versatile microwave oven. It offers quick and convenient cooking options	1290
205	Air Conditioner	Stay cool and comfortable with our powerful air conditioner. Enjoy adjustable settings, energy efficiency, and a refreshing atmosphere, even on the hottest days.	4990
206	Blender	Create delicious smoothies and culinary delights with our high-performance blender. Its durable design and multiple speed settings make blending a joy.	790
207	Vacuum Cleaner	Keep your home spotless with our efficient vacuum cleaner. Featuring powerful suction and versatile attachments, it makes cleaning effortless.	1990
208	Electric Kettle	Boil water quickly and efficiently with our electric kettle. Enjoy safety features, sleek design, and convenience for your daily hot beverages.	4900
209	Toaster	Start your mornings right with our reliable toaster. Whether it's crispy toast or warm bagels, enjoy even toasting and easy operation every time.	3900
300	Hair Dryer	Achieve salon-quality hair drying with our powerful hair dryer. Fast drying, multiple heat settings, and ergonomic design for effortless styling.	5900

Orders

OrderID	CustomerID	OrderDate	TotalAmount
300	1	2023-12-01	89900
301	2	2023-12-02	7990
302	3	2023-12-03	9990
303	4	2023-12-04	2290
304	5	2023-12-05	5990
305	6	2023-12-06	890
306	7	2023-12-07	2990
307	8	2023-12-08	5900
308	9	2023-12-09	4900
309	10	2023-12-10	6900

OrderDetails

OrderDetailsID	OrderID	ProductID	Quantity
400	300	200	1
401	301	202	5
402	302	203	7
403	303	204	1
404	304	205	2
405	305	206	5
406	306	207	6
407	307	208	1
408	308	209	2
409	309	300	2

Inventory

InventoryID	ProductID	QuantityInStock	LastStockUpdate
500	200	1	2023-12-01
501	202	1	2023-12-02
502	203	1	2023-12-03
503	204	0	2023-12-04
504	205	1	2023-12-05
505	206	0	2023-12-06
506	207	1	2023-12-07
507	208	1	2023-12-08
508	209	0	2023-12-09
509	300	0	2023-12-10

Task - 2 - SELECT , WHERE , AND , LIKE

1) Write an sql query to retrieve the names and emails of all customers.

Firstname	LastName	Email
John	Doe	john@example.com
Alice	Smith	alice@example.com
Bob	Johnson	bob@example.com
Emily	Brown	emily@example.com
Michael	Davis	michael@example.com
Sarah	Wilson	sarah@example.com
David	Miller	david@example.com
Olivia	Garcia	olivia@example.com
Sophia	Martinez	sophia@example.com
James	Thompson	james@example.com

2) Write an SQL query to list all orders with their order dates and corresponding customer names.

Name	OrderDate
John Doe	2023-12-01
Alice Smith	2023-12-02
Bob Johnson	2023-12-03
Emily Brown	2023-12-04
Michael Davis	2023-12-05
Sarah Wilson	2023-12-06
David Miller	2023-12-07
Olivia Garcia	2023-12-08
Sophia Martinez	2023-12-09
James Thompson	2023-12-10

3) Write an SQL query to insert a new customer record into the “Customers” table . Include customer information such as name , email , and address.

```
mysql> insert into customers values(11,'sheldon', 'Cooper','bazinga@gmail.com',1734467891,'123 Main St');
Query OK, 1 row affected (0.03 sec)
```

4) Write a SQL query to update the price of all electronic gadgets in the “Products” table By increasing them by 10%.

```
mysql> update products set price=price*1.1;
Query OK, 10 rows affected (0.03 sec)
Rows matched: 10  Changed: 10  Warnings: 0
```

5) Write a SQl query to update the contact information of the specific customer in the “Customers” table .Allow users to input the customer ID and new contact information.

```
mysql> update customers set  Email  = "Doe@example.com", Address = '121 Down street' where CustomerID  = 1;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

6) Write a specific query to delete a specific order and its associated order details from the “Orders” and “OrderDetails” table .

```
mysql> alter table orderdetails
    -> add constraint foreign_key
    -> foreign key(OrderID)references orders(orderid) on delete cascade;
Query OK, 9 rows affected (0.17 sec)
Records: 9  Duplicates: 0  Warnings: 0

mysql> delete from orderDetails where OrderID  = 301;
Query OK, 1 row affected (0.01 sec)
```

7) Write a specific query to delete all orders and their associated order details from the “Orders” and “OrderDetails” table for a specific customer .

```
mysql> DELETE orders
    -> FROM orders
    -> JOIN orderdetails  on orders.orderID  = orderDetails.orderID
    -> WHERE orders.CustomerID  = 6;
Query OK, 1 row affected (0.02 sec)
```

8) Write a SQL query to insert a new order into the “Orders”table .Include the Customer ID , order date , and any other necessary information.

```
mysql> insert into Orders values(310,11,'2023-12-10', 1230);
Query OK, 1 row affected (0.03 sec)
```

9) Write a SQL query to recalculate and update the total cost of each order in “Orders” table based on the prices and quantities in the “OrderDetails”table .

```
mysql>  UPDATE Orders INNER JOIN OrderDetails
    -> ON Orders.OrderID = OrderDetails.OrderID
    -> SET Orders.TotalAmount = OrderDetails.Quantity*Orders.TotalAmount;
Query OK, 7 rows affected (0.02 sec)
```

10) Write a SQL query to insert a new electronic gadget product into the “Products” table,including product name , category , price , and any other relevant details.

```
mysql> INSERT INTO Products (ProductID, ProductName, Description, Price) VALUES
   -> (24, 'Smartwatch', 'Enhance your lifestyle with our smartwatch. Stay connected, track fitness, and manage your day with style a
nd convenience.', 199);
Query OK, 1 row affected (0.02 sec)
```

11) Write a SQL query to update the status of a specific order in the “Orders” table (e.g., from pending to shipped).

```
mysql> update orders set status  = 'shipped' where orderID  = 305;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0
```

Task - 3 AGGREGATE FUNCTIONS , HAVING,ORDER BY , GROUP BY

1) Write an SQL query to retrieve a list of all orders along with customer information (e.g., customer name) for each order.

OrderID	OrderDate	TotalAmount	FirstName	LastName
301	2023-12-02	24968750	Alice	Smith
302	2023-12-03	167901930	Bob	Johnson
303	2023-12-04	2290	Emily	Brown
304	2023-12-05	191680	Michael	Davis
306	2023-12-07	23250240	David	Miller
307	2023-12-08	5900	Olivia	Garcia
308	2023-12-09	156800	Sophia	Martinez
309	2023-12-10	220800	James	Thompson
310	2023-12-10	1230	sheldon	Cooper

2) Write an SQL query to find the total revenue generated by each electronic gadget product include the product name and the total revenue.

TotalAmount	product_Name
167901930	Refrigerator
2290	Microwave Oven
191680	Air Conditioner
23250240	Vacuum Cleaner
5900	Electric Kettle
156800	Toaster
220800	Hair Dryer

3) Write an SQL query to list all customers who have made at least one purchase .Include their name and contact information.

FirstName	LastName	Email	phone
Alice	Smith	alice@example.com	1876543210
Bob	Johnson	bob@example.com	1551234567
Emily	Brown	emily@example.com	1779998888
Michael	Davis	michael@example.com	1445556666
David	Miller	david@example.com	1887776666
Olivia	Garcia	olivia@example.com	1661112222
Sophia	Martinez	sophia@example.com	1334445555
James	Thompson	james@example.com	1990001111
sheldon	Cooper	bazinga@gmail.com	1734467891

4) Write a SQL query to find the most popular electronic gadgets , which is one with the highest total quantity ordered.Include the product name and total quantity ordered.

productName	quantity
Refrigerator	7

5) Write an SQL query to calculate the average order value for each customer. Include the customer's name and their average order value.

Customer_name	average
Bob Johnson	23985990.00
Emily Brown	2290.00
Michael Davis	95840.00
David Miller	3875040.00
Olivia Garcia	5900.00
Sophia Martinez	78400.00
James Thompson	110400.00

6) Write an SQL query to find the order with the highest total revenue. Include the order ID, customer information, and the total revenue.

OrderID	TotalAmount	Customer_name	Email	Phone
302	167901930	Bob Johnson	bob@example.com	1551234567

7) Write an SQL query to list electronic gadgets and the number of times each product has been ordered.

Product Name	Quantity Ordered
Refrigerator	7
Microwave Oven	1
Air Conditioner	2
Vacuum Cleaner	6
Electric Kettle	1
Toaster	2
Hair Dryer	2

8) Write an SQL query to find customers who have purchased a specific electronic gadget product. Allow users to input the product name as a parameter.

product_name	Customer name
Refrigerator	Bob Johnson
Microwave Oven	Emily Brown
Air Conditioner	Michael Davis
Vacuum Cleaner	David Miller
Electric Kettle	Olivia Garcia
Toaster	Sophia Martinez
Hair Dryer	James Thompson

9) Write an SQL query to calculate the total revenue generated by all orders placed within a specific time period. Allow users to input the start and end dates as parameters.
(2023-12-04 to 2023-12-09)

Total revenue
23444210

TASK 4. SUBQUERY AND IT'S TYPE

1) Write an SQL query to find out which customers have not placed any orders.

CustomerID	FirstName	LastName
1	John	Doe
6	Sarah	Wilson

2) Write an SQL query to calculate the total revenue generated by TechShop.

TotalRevenue
216699620

3) Write an SQL query to find the total number of products available for sale.

TotalProducts
11

4) Write an SQL query to calculate the average quantity ordered for products in a specific category. Allow users to input the category name as a parameter.

Product Name	average
Microwave Oven	1419.00

5) Write an SQL query to calculate the total revenue generated by a specific customer.

TotalRevenue
13134

6) Write an SQL query to find the customers who have placed the most orders.

max quantity	Customer name
7	Bob Johnson

7) Write an SQL query to find the most popular product category, which is the one with the highest total quantity ordered across all orders.

ProductID	ProductName	TotalQuantityOrdered
203	Refrigerator	7

8) Write an SQL query to find the customer who has spent the most money (highest total revenue) on electronic gadgets. List their name and total spending.

FirstName	LastName	TotalSpending
Alice	Smith	24968750

9) Write an SQL query to calculate the average order value (total revenue divided by the number of orders) for all customers.

AverageOrderValue
2787964.8889

10) Write an SQL query to find the total number of orders placed by each customer and list their names along with the order count.

customer name	Quantity ordered
Bob Johnson	7
Emily Brown	1
Michael Davis	2
David Miller	6
Olivia Garcia	1
Sophia Martinez	2
James Thompson	2

Assignments-2

A simplified Student Information System (SIS) database. The SIS database contains information about students, courses, and enrollments. Task is to perform various SQL operations on this database to retrieve and manipulate data.

TASK - 1 DATABASE DESIGN

1) Create the database named 'SISDB'

```
mysql> CREATE DATABASE SISDB;
Query OK, 1 row affected (0.03 sec)
```

,

2) Define the schema for the students,Courses,Enrollment , Teachers and Payments table based on the provided schema .Write the SQLscript to create the mentioned tables with appropriate data type , constraints and relationship.

Students

```
mysql>  CREATE TABLE Students(
    -> student_id INT PRIMARY KEY,
    ->  first_name VARCHAR(25),
    ->  last_name VARCHAR(25),
    ->  date_of_birth DATE,
    ->  email VARCHAR(50),
    ->  phone_number BIGINT
    -> );
```

Courses

```
mysql> CREATE TABLE Courses(
    -> course_id INTEGER PRIMARY KEY,
    ->  course_name VARCHAR(50),
    ->  credits VARCHAR(50),
    ->  teacher_id INTEGER ,
    -> FOREIGN KEY(teacher_id) REFERENCES Teacher(teacher_id)
    -> );
```

Enrollments

```
mysql> CREATE TABLE Enrollments(
    -> enrollment_id INT PRIMARY KEY,
    -> student_id INT ,
    -> course_id INT,
    -> enrollment_date DATE,
    -> FOREIGN KEY(student_id) REFERENCES Students(student_id),
    ->  FOREIGN KEY(course_id) REFERENCES Courses(course_id)
    -> );
```

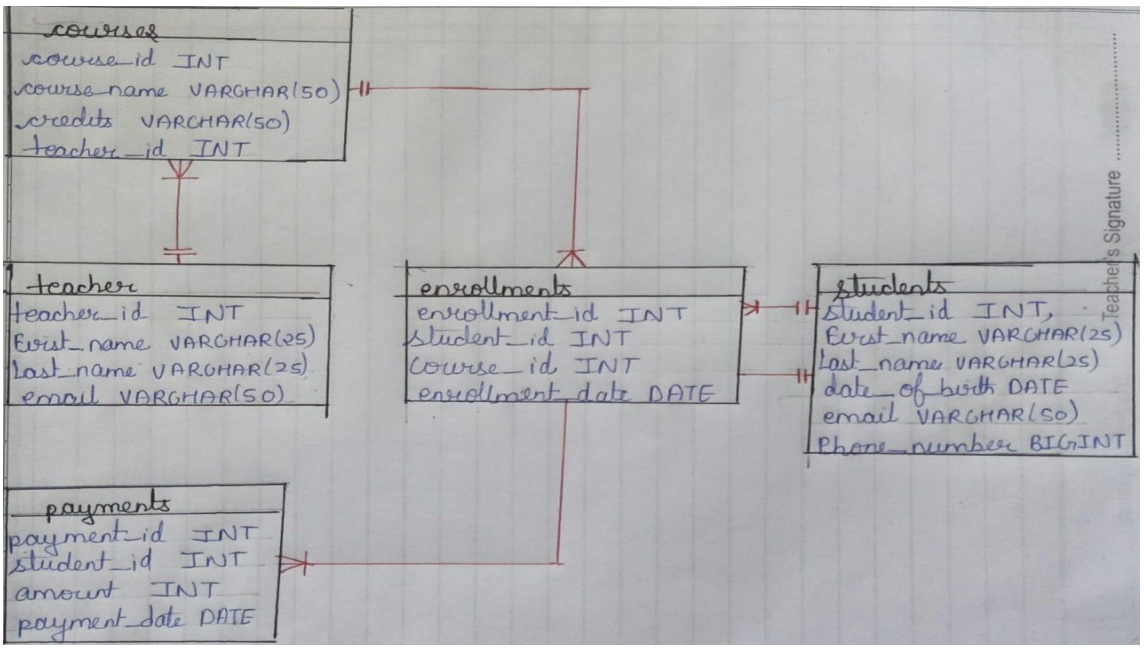
Teacher

```
mysql> CREATE TABLE Teacher(
    -> teacher_id INT PRIMARY KEY,
    -> first_name VARCHAR(25),
    -> last_name VARCHAR(25),
    -> email VARCHAR(50)
    -> );
```

Payments

```
ysql> CREATE TABLE Payments(
   -> payment_id INT PRIMARY KEY,
   -> student_id INT ,
   -> amount INTEGER,
   -> payment_date DATE,
   -> FOREIGN KEY(student_id) REFERENCES Students(student_id)
   -> );
```

3) create E-R(entity relationship) diagram.



4) Insert at least 10 sample record into each of the following tables .

Students

student_id	first_name	last_name	date_of_birth	email	phone_number
1	John	Doe	2000-05-15	john@example.com	9234567890
2	Jane	Smith	2001-08-25	jane@example.com	9876543210
3	Alice	Johnson	1999-11-03	alice@example.com	9551112222
4	Bob	Anderson	2002-04-18	bob@example.com	9998887777
5	Eva	Garcia	2003-09-21	eva@example.com	8443332222
6	Michael	Brown	2000-12-08	michael@example.com	7776665555
7	Sophia	Lee	2001-07-14	sophia@example.com	9223334444
8	Oliver	Martinez	1998-02-28	oliver@example.com	7667778888
9	sheldon	cooper	1998-01-12	bazinga@gamil.com	7574832312
10	racheal	green	1999-07-10	racheal@gamil.com	9574432215

Courses

course_id	course_name	credits	teacher_id
301	C and C++	4	101
302	Core Java	3	102
303	Python	3	103
304	C#	4	104
305	Mysql	5	105
306	JavaScript	3	106
307	PHP	4	107
308	Kotlin	4	108
309	MariaDB	3	109
400	PostgreSQL	4	110

Enrollments

enrollment_id	student_id	course_id	enrollment_date
501	1	301	2023-09-01
502	2	302	2023-09-03
503	3	303	2023-09-05
504	4	304	2023-09-07
505	5	305	2023-09-09
506	6	306	2023-09-11
507	7	307	2023-09-13
508	8	308	2023-09-15
509	9	309	2023-09-17
600	10	400	2023-09-19

Teacher

teacher_id	first_name	last_name	email
101	Emily	Johnson	emily@example.com
102	Michael	Smith	michael@example.com
103	Sophia	Williams	sophia@example.com
104	Daniel	Brown	daniel@example.com
105	Olivia	Miller	olivia@example.com
106	David	Wilson	david@example.com
107	Emma	Anderson	emma@example.com
108	William	Martinez	william@example.com
109	Samantha	Garcia	samantha@example.com
110	Aiden	Lopez	aiden@example.com

Payments

payment_id	student_id	amount	payment_date
901	1	500	2023-10-01
902	2	750	2023-10-03
903	3	600	2023-10-05
904	4	900	2023-10-07
905	5	400	2023-10-09
906	6	550	2023-10-11
907	7	800	2023-10-13
908	8	700	2023-10-15
909	9	950	2023-10-17
1000	10	350	2023-10-19

5) Create appropriate Primary Key and Foreign Key constraints for referential integrity.

Primary key

TABLE_NAME	CONSTRAINT_NAME
teacher	PRIMARY
students	PRIMARY
courses	PRIMARY
enrollments	PRIMARY
payments	PRIMARY

Foreign Key

TABLE_NAME	CONSTRAINT_NAME
courses	courses_ibfk_1
enrollments	enrollments_ibfk_1
enrollments	enrollments_ibfk_2
payments	payments_ibfk_1

TASK - 2 SELECT , WHERE , BETWEEN , AND , LIKE

1) Write a SQL query to insert a new Student into the"Students" table with the following criteria.

```
ysql> insert into students values(11,'John','Doe','1995-08-15','Doe@example.com','123456780');
uery OK, 1 row affected (0.03 sec)
```

2) Write a SQl query to enroll a student in a course .Choose an existing student and course and insert a record into the "Enrollment" table with enrollment date.

```
mysql> insert into enrollments values(601,1,304,'2023-12-10');
Query OK, 1 row affected (0.02 sec)
```

3) Update the email address of a specific teacher in a "Teacher" table.Choose any teacher and modify their email address.

```
mysql> update teacher set email = "johnson@example.com" where teacher_id  = 101;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

4) write a SQl query to delete a specific enrollment record from the "Enrollments" table.Select an enrollment record based on the student and course.

```
mysql> delete from enrollments where student_id  = 1 and course_id  = 304;
Query OK, 1 row affected (0.04 sec)
```

5) Update the “Courses” table to assign a specific teacher to a course .Choose any course and teacher from the respective table .

```
ysql> update Courses set teacher_id  = 110 where course_id  = 305;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

6) Update the payment amount for a specific payment record in the “Payments” table .Choose any payment record and modify the payment amount.

```
ysql> update payments set amount  = 400 where payment_id  = 1000;
uery OK, 1 row affected (0.02 sec)
ows matched: 1  Changed: 1  Warnings: 0
```

7) Delete a specific student from the "Students" table and remove all their enrollment records from the "Enrollments" table. Be sure to maintain referential integrity.

```
mysql> delete enrollments
    -> from enrollments
    -> join students on students.student_id  = enrollments.student_id where enrollments.student_id = 2;
Query OK, 1 row affected (0.04 sec)
```

TASK 3 - AGGREGATE FUNCTIONS, HAVING, ORDER BY, GROUP BY and JOINS:

1) Write an SQL query to calculate the total payments made by a specific student. You will need to join the "Payments" table with the "Students" table based on the student's ID.

first_name	last_name	total_payments
Bob	Anderson	900

2) Write an SQL query to retrieve a list of courses along with the count of students enrolled in each course. Use a JOIN operation between the "Courses" table and the "Enrollments" table.

course_id	course_name	student_count
301	C and C++	1
302	Core Java	0
303	Python	1
304	C#	1
305	Mysql	1
306	JavaScript	1
307	PHP	1
308	Kotlin	1
309	MariaDB	1
400	PostgreSQL	1

3) Write an SQL query to find the names of students who have not enrolled in any course. Use a LEFT JOIN between the "Students" table and the "Enrollments" table to identify students without enrollments.

first_name	last_name
Jane	Smith
John	Doe

4) Write an SQL query to retrieve the first name, last name of students, and the names of the courses they are enrolled in. Use JOIN operations between the "Students" table and the "Enrollments" and "Courses" tables.

first_name	last_name	course_name
John	Doe	C and C++
Alice	Johnson	Python
Bob	Anderson	C#
Eva	Garcia	Mysql
Michael	Brown	JavaScript
Sophia	Lee	PHP
Oliver	Martinez	Kotlin
sheldon	cooper	MariaDB
racheal	green	PostgreSQL

5) Create a query to list the names of teachers and the courses they are assigned to. Join the "Teacher" table with the "Courses" table.

first_name	last_name	course_name
John	Doe	C and C++
Alice	Johnson	Python
Bob	Anderson	C#
Eva	Garcia	Mysql
Michael	Brown	JavaScript
Sophia	Lee	PHP
Oliver	Martinez	Kotlin
sheldon	cooper	MariaDB
racheal	green	PostgreSQL

6) Retrieve a list of students and their enrollment dates for a specific course. You'll need to join the "Students" table with the "Enrollments" and "Courses" tables.

first_name	last_name	enrollment_date
Oliver	Martinez	2023-09-15

7) Find the names of students who have not made any payments. Use a LEFT JOIN between the "Students" table and the "Payments" table and filter for students with NULL payment records.

first_name	last_name
John	Doe

8) Write a query to identify courses that have no enrollments. You'll need to use a LEFT JOIN between the "Courses" table and the "Enrollments" table and filter for courses with NULL enrollment records.

course_id	course_name
302	Core Java

9) Identify students who are enrolled in more than one course. Use a self-join on the "Enrollments" table to find students with multiple enrollment records.

```
mysql> SELECT DISTINCT e1.student_id, s.first_name, s.last_name
    -> FROM Enrollments e1
    -> JOIN Enrollments e2 ON e1.student_id = e2.student_id AND e1.course_id <> e2.course_id
    -> JOIN Students s ON e1.student_id = s.student_id;
Empty set (0.00 sec)
```

10) Find teachers who are not assigned to any courses. Use a LEFT JOIN between the "Teacher" table and the "Courses" table and filter for teachers with NULL course assignments.

teacher_id	first_name	last_name
105	Olivia	Miller

TASK -4 SUBQUERY AND IT'S TYPES

1) Write an SQL query to calculate the average number of students enrolled in each course. Use aggregate functions and subqueries to achieve this.

average_students_per_course
1.0000

2) Identify the student(s) who made the highest payment. Use a subquery to find the maximum payment amount and then retrieve the student(s) associated with that amount.

student_id	first_name	last_name	highest_payment_amount
9	sheldon	cooper	950

3) Retrieve a list of courses with the highest number of enrollments. Use subqueries to find the course(s) with the maximum enrollment count.

course_id	course_name	max_enroll_count
301	C and C++	1
303	Python	1
304	C#	1
305	Mysql	1
306	JavaScript	1
307	PHP	1
308	Kotlin	1
309	MariaDB	1
400	PostgreSQL	1

4) Calculate the total payments made to courses taught by each teacher. Use subqueries to sum payments for each teacher's courses.

teacher_id	first_name	last_name	total_payments
101	Emily	Johnson	500
102	Michael	Smith	0
103	Sophia	Williams	600
104	Daniel	Brown	900
105	Olivia	Miller	0
106	David	Wilson	550
107	Emma	Anderson	800
108	William	Martinez	700
109	Samantha	Garcia	950
110	Aiden	Lopez	800

5) Identify students who are enrolled in all available courses. Use subqueries to compare a student's enrollments with the total number of courses.

```
Empty set (0.00 sec)
```

6) Retrieve the names of teachers who have not been assigned to any courses. Use subqueries to find teachers with no course assignments.

teacher_id	first_name	last_name
105	Olivia	Miller

7) Calculate the average age of all students. Use subqueries to calculate the age of each student based on their date of birth.

average_age
23.3636

8) Identify courses with no enrollments. Use subqueries to find courses without enrollment record.

course_id	course_name
302	Core Java

9) Calculate the total payments made by each student for each course they are enrolled in. Use subqueries and aggregate functions to sum payments.

student_id	course_id	total_payments
1	301	500
3	303	600
4	304	900
5	305	400
6	306	550
7	307	800
8	308	700
9	309	950
10	400	400

10) Identify students who have made more than one payment. Use subqueries and aggregate functions to count payments per student and filter for those with counts greater than one.

```
Empty set (0.00 sec)
```

11) Write an SQL query to calculate the total payments made by each student. Join the "Students" table with the "Payments" table and use GROUP BY to calculate the sum of payments for each student.

student_id	first_name	last_name	total_payments
1	John	Doe	500
2	Jane	Smith	750
3	Alice	Johnson	600
4	Bob	Anderson	900
5	Eva	Garcia	400
6	Michael	Brown	550
7	Sophia	Lee	800
8	Oliver	Martinez	700
9	sheldon	cooper	950
10	racheal	green	400
11	John	Doe	0

12) Retrieve a list of course names along with the count of students enrolled in each course. Use JOIN operations between the "Courses" table and the "Enrollments" table and GROUP BY to count enrollments.

course_name	student_count
C and C++	1
Core Java	0
Python	1
C#	1
Mysql	1
JavaScript	1
PHP	1
Kotlin	1
MariaDB	1
PostgreSQL	1

13) Calculate the average payment amount made by students. Use JOIN operations between the "Students" table and the "Payments" table and GROUP BY to calculate the average.

student_id	first_name	last_name	average_payment
1	John	Doe	500.0000
2	Jane	Smith	750.0000
3	Alice	Johnson	600.0000
4	Bob	Anderson	900.0000
5	Eva	Garcia	400.0000
6	Michael	Brown	550.0000
7	Sophia	Lee	800.0000
8	Oliver	Martinez	700.0000
9	sheldon	cooper	950.0000
10	racheal	green	400.0000
11	John	Doe	NULL

Assignment - 3

An advanced banking system that includes various types of accounts, such as savings and current accounts. The system should support account creation, deposits, withdrawals, and interest calculations.

TASK-1 DATABASE DESIGN

1) Create a database named “HMBank”.

```
mysql> create database HMBank;
Query OK, 1 row affected (0.03 sec)
```

2) Define the Schema for Customers , Accounts and Transactions tables based on the provided schema.

Customers

```
mysql> CREATE TABLE Customers(
    -> customer_id INT PRIMARY KEY,
    -> first_name VARCHAR(25),
    ->  last_name VARCHAR(25),
    -> DOB DATE,
    -> email VARCHAR(50),
    ->  phone_number BIGINT,
    ->  address TEXT
    -> );
Query OK, 0 rows affected (0.09 sec)
```

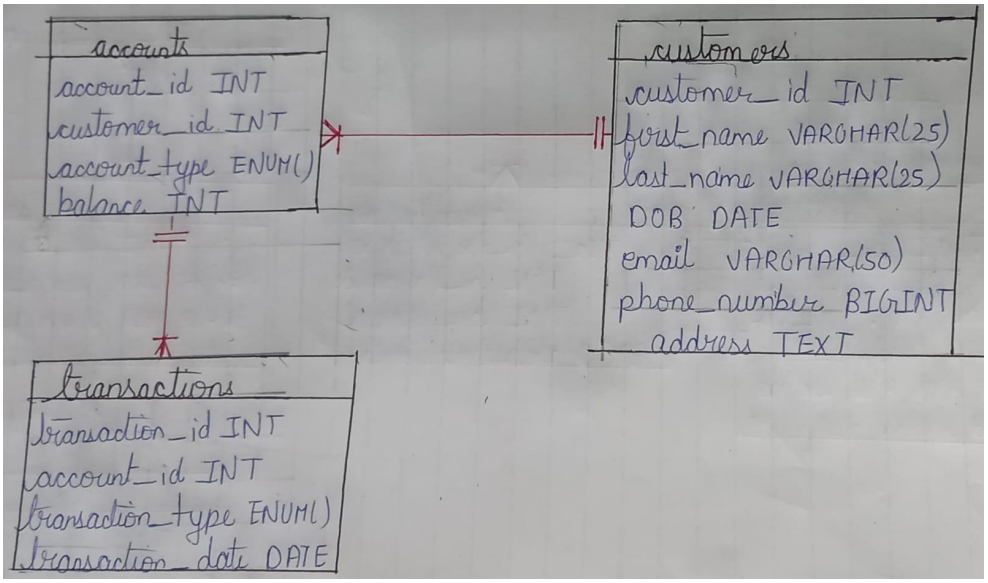
Accounts

```
ysql> CREATE TABLE Accounts(
    -> account_id INT PRIMARY KEY,
    -> customer_id INT,
    -> account_type  ENUM('savings','current','zero_balance'),
    -> balance INT,
    -> FOREIGN KEY(customer_id) REFERENCES customers(customer_id)
    -> );
uery OK, 0 rows affected (0.09 sec)
```

Transactions

```
ysql>  CREATE TABLE Transactions(
    -> transaction_id INT PRIMARY KEY,
    -> account_id INT,
    -> transcation_type ENUM('deposit','withdrawal','transfer'),
    -> amount INT,
    -> transaction_date DATE,
    -> FOREIGN KEY(account_id) REFERENCES Accounts(account_id)
    -> );
uery OK, 0 rows affected (0.09 sec)
```

4) Create an E-R(entity relationship diagram) for the database .



5) Create appropriate Primary key and Foreign key constraint for referential integrity.

Primary key

TABLE_NAME	CONSTRAINT_NAME
customers	PRIMARY
accounts	PRIMARY
transactions	PRIMARY

Foreign key

TABLE_NAME	CONSTRAINT_NAME
accounts	accounts_ibfk_1
transactions	transactions_ibfk_1

6) Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

Accounts

Field	Type	Null	Key	Default
account_id	int	NO	PRI	NULL
customer_id	int	YES	MUL	NULL
account_type	enum('savings','current','zero_balance')	YES		NULL
balance	int	YES		NULL

Customers

Field	Type	Null	Key	Default
customer_id	int	NO	PRI	NULL
first_name	varchar(25)	YES		NULL
last_name	varchar(25)	YES		NULL
DOB	date	YES		NULL
email	varchar(50)	YES		NULL
phone_number	bigint	YES		NULL
address	text	YES		NULL

Transactions

Field	Type	Null	Key	Default
transaction_id	int	NO	PRI	NULL
account_id	int	YES	MUL	NULL
transaction_type	enum('deposit','withdrawal','transfer')	YES		NULL
amount	int	YES		NULL
transaction_date	date	YES		NULL

TASK-2 SELECT , WHERE , BETWEEN , AND , LIKE

1) Insert at least 10 sample records into each of the following tables.

Customers

customer_id	first_name	last_name	DOB	email	phone_number	address
1	Alice	Smith	1995-08-20	alice@example.com	9234567890	1234 Elm Street
2	Bob	Johnson	1988-04-10	bob@example.com	9876543210	5678 Oak Avenue
3	Emily	Brown	1992-12-05	emily@example.com	5554443333	910 Cedar Road
4	Daniel	Miller	1985-06-25	daniel@example.com	1112223333	6789 Pine Lane
5	Sophia	Davis	1998-09-14	sophia@example.com	9998887777	3210 Maple Street
6	Michael	Wilson	1990-02-28	michael@example.com	7776665555	4567 Birch Avenue
7	Olivia	Martinez	1994-07-03	olivia@example.com	2223334444	7890 Ash Street
8	William	Taylor	1987-11-19	william@example.com	4445556666	2345 Pinecrest Drive
9	Ava	Anderson	1996-03-22	ava@example.com	8889991111	8765 Cedar Lane
10	James	Garcia	1984-10-08	james@example.com	6667778888	5432 Elm Avenue

Accounts

account_id	customer_id	account_type	balance
101	1	savings	5000
102	2	current	10000
103	3	savings	7500
104	4	zero_balance	0
105	5	current	5500
106	6	savings	12000
107	7	current	9300
108	8	zero_balance	0
109	9	current	2000
110	10	savings	9000

Transactions

transaction_id	account_id	transaction_type	amount	transaction_date
501	101	deposit	1000	2023-01-15
502	101	withdrawal	500	2023-02-03
503	102	deposit	2000	2023-02-10
504	103	transfer	1500	2023-03-20
505	104	withdrawal	800	2023-04-05
506	102	transfer	3000	2023-04-18
507	103	deposit	1200	2023-05-01
508	104	withdrawal	1000	2023-06-10
509	105	deposit	500	2023-07-15
600	105	withdrawal	200	2023-07-25

2) Write SQL query for the following tasks

1) Write a SQL query to increase the balance of a specific account by a certain amount.

```
mysql> update accounts set balance  = balance+50 where customer_id  = 5;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

2) Write a SQL query to combine the first and last name of customers as full_name.

full_name
Alice Smith
Bob Johnson
Emily Brown
Daniel Miller
Sophia Davis
Michael Wilson
Olivia Martinez
William Taylor
Ava Anderson
James Garcia

3) Write a SQl query to remove accounts with a balance of 0 where account type is saving

```
mysql> delete from accounts where account_type  = 'savings' and balance  = 0;
Query OK, 0 rows affected (0.00 sec)
```

4) Write a SQL query to find customers living in a specific city.

first_name	last_name	address
Daniel	Miller	6789 Pine Lane

5) Write a SQL query to get the account balance for a specific account.

balance
9000

6) Write a SQl query to List all current accounts with a balance greater than 1000.

account_id	customer_id	account_type	balance
101	1	savings	5000
102	2	current	10000
103	3	savings	7500
105	5	current	5550
106	6	savings	12000
107	7	current	9300
109	9	current	2000
110	10	savings	9000

7) Write a SQL query to retrieve all transactions for a specific account.

account_id	customer_id	account_type	balance
101	1	savings	5000
102	2	current	10000
103	3	savings	7500
105	5	current	5550
106	6	savings	12000
107	7	current	9300
109	9	current	2000
110	10	savings	9000

8) Write a SQL query to list all transaction corresponding customer.

concat(c.first_name,' ',c.last_name)	transaction_type	amount	transaction_date
Alice Smith	deposit	1000	2023-01-15
Alice Smith	withdrawal	500	2023-02-03
Bob Johnson	deposit	2000	2023-02-10
Emily Brown	transfer	1500	2023-03-20
Daniel Miller	withdrawal	800	2023-04-05
Bob Johnson	transfer	3000	2023-04-18
Emily Brown	deposit	1200	2023-05-01
Daniel Miller	withdrawal	1000	2023-06-10
Sophia Davis	deposit	500	2023-07-15
Sophia Davis	withdrawal	200	2023-07-25

9) Write a SQL query to retrieve the name, account type and email of all customers.

first_name	last_name	account_type	email
Alice	Smith	savings	alice@example.com
Bob	Johnson	current	bob@example.com
Emily	Brown	savings	emily@example.com
Daniel	Miller	zero_balance	daniel@example.com
Sophia	Davis	current	sophia@example.com
Michael	Wilson	savings	michael@example.com
Olivia	Martinez	current	olivia@example.com
William	Taylor	zero_balance	william@example.com
Ava	Anderson	current	ava@example.com
James	Garcia	savings	james@example.com

10) Write a SQL query to Calculate the interest accrued on savings accounts based on a given interest rate.

Interest rate = 3%

account_id	initial_balance	accrued_interest
101	5000	150.00
103	7500	225.00
106	12000	360.00
110	9000	270.00

11) Write a SQL query to Identify accounts where the balance is less than a specified overdraft limit.

1000 is considered as the assumed overdraft limit

account_id	initial_balance	accrued_interest
101	5000	150.00
103	7500	225.00
106	12000	360.00
110	9000	270.00

12) Write a SQL query to Find customers not living in a specific city.

City = “New York”

customer_id	first_name	last_name	address
1	Alice	Smith	1234 Elm Street
2	Bob	Johnson	5678 Oak Avenue
3	Emily	Brown	910 Cedar Road
4	Daniel	Miller	6789 Pine Lane
5	Sophia	Davis	3210 Maple Street
6	Michael	Wilson	4567 Birch Avenue
7	Olivia	Martinez	7890 Ash Street
8	William	Taylor	2345 Pinecrest Drive
9	Ava	Anderson	8765 Cedar Lane
10	James	Garcia	5432 Elm Avenue

Task 3. AGGREGATE FUNCTIONS, HAVING, ORDER BY, GROUP BY and JOINS:

1) Write a SQL query to Find the average account balance for all customers .

average_balance
6035.0000

2) Write a SQL query to Retrieve the top 10 highest account balances.

transaction_id	account_id	transaction_type	amount	transaction_date
506	102	transfer	3000	2023-04-18
503	102	deposit	2000	2023-02-10
504	103	transfer	1500	2023-03-20

3) Write a SQL query to Calculate Total Deposits for All Customers in specific date.

Date = 2023-04-18

total_deposits
2000

4) Write a SQL query to Find the Oldest and Newest Customers.

first_name	last_name	date_of_birth	customer_type
James	Garcia	1984-10-08	Oldest
Sophia	Davis	1998-09-14	Newest

5) Write a SQL query to Retrieve transaction details along with the account type.

transaction_id	account_type	transaction_type	amount	transaction_date
501	savings	deposit	1000	2023-01-15
502	savings	withdrawal	500	2023-02-03
503	current	deposit	2000	2023-02-10
504	savings	transfer	1500	2023-03-20
505	zero_balance	withdrawal	800	2023-04-05
506	current	transfer	3000	2023-04-18
507	savings	deposit	1200	2023-05-01
508	zero_balance	withdrawal	1000	2023-06-10
509	current	deposit	500	2023-07-15
600	current	withdrawal	200	2023-07-25

6) Write a SQL query to Get a list of customers along with their account details.

customer_id	first_name	last_name	email	phone_number	address	account_id	account_type	balance
1	Alice	Smith	alice@example.com	9234567890	1234 Elm Street	101	savings	500
2	Bob	Johnson	bob@example.com	9876543210	5678 Oak Avenue	102	current	1000
3	Emily	Brown	emily@example.com	5554443333	910 Cedar Road	103	savings	700
4	Daniel	Miller	daniel@example.com	1112223333	6789 Pine Lane	104	zero_balance	0
5	Sophia	Davis	sophia@example.com	9998887777	3210 Maple Street	105	current	550
6	Michael	Wilson	michael@example.com	7776665555	4567 Birch Avenue	106	savings	1200
7	Olivia	Martinez	olivia@example.com	2223334444	7890 Ash Street	107	current	900
8	William	Taylor	william@example.com	4445556666	2345 Pinecrest Drive	108	zero_balance	0
9	Ava	Anderson	ava@example.com	8889991111	8765 Cedar Lane	109	current	200
10	James	Garcia	james@example.com	6667778888	5432 Elm Avenue	110	savings	900

7) Write a SQL query to Retrieve transaction details along with customer information for a specific account.

name	email	phone	id	type	amount	date
Sophia Davis	sophia@example.com	9998887777	509	deposit	500	2023-07-15
Sophia Davis	sophia@example.com	9998887777	600	withdrawal	200	2023-07-25

8) Write a SQL query to Identify customers who have more than one account.

```
ysql> select customer_id,COUNT(account_id) AS accounts from  Accounts group by customer_id having COUNT(account_id) > 1;
mpty set (0.00 sec)
```

9) Write a SQL query to Calculate the difference in transaction amounts between deposits and withdrawals.

difference
2200

10) Write a SQL query to Calculate the average daily balance for each account over a specified period.

account_id	average_daily_balance
101	5000.0000
101	5000.0000
102	10000.0000
103	7500.0000

11) Calculate the total balance for each account type.

account_type	total_balance
savings	33500
current	26850
zero_balance	0

12) Identify accounts with the highest number of transactions order by descending order.

account_id	num_transactions
101	2
102	2
103	2
104	2
105	2

13) List customers with high aggregate account balances, along with their account types.

customer_id	first_name	last_name	account_type	total_balance
6	Michael	Wilson	savings	12000
2	Bob	Johnson	current	10000
7	Olivia	Martinez	current	9300
10	James	Garcia	savings	9000
3	Emily	Brown	savings	7500
5	Sophia	Davis	current	5550
1	Alice	Smith	savings	5000
9	Ava	Anderson	current	2000
4	Daniel	Miller	zero_balance	0
8	William	Taylor	zero_balance	0

14) Identify and list duplicate transactions based on transaction amount, date, and account.

```
mysql> SELECT account_id, amount,transaction_date,COUNT(*) AS duplicate_count FROM Transactions GROUP BY account_id, amount, transact
ion_date HAVING COUNT(*) > 1;
Empty set (0.00 sec)
```

TASK -4 SUBQUERY AND IT'S TYPES

1) Retrieve the customer(s) with the highest balance.

customer_id	first_name	last_name	max_balance
6	Michael	Wilson	12000

2) Calculate the average account balance for customers who have more than one account.

```
mysql> SELECT c.customer_id,AVG(a.balance) AS average_balance FROM Customers c  JOIN Accounts a ON c.customer_id = a.customer_id WHERE
 c.customer_id IN ( SELECT customer_id FROM Accounts GROUP BY customer_id HAVING COUNT(account_id) > 1) GROUP BY c.customer_id;
Empty set (0.00 sec)
```

3) Retrieve accounts with transactions whose amounts exceed the average transaction amount.

account_id	transaction_id	amount	avg_transaction_amount
102	503	2000	1170.0000
103	504	1500	1170.0000
102	506	3000	1170.0000
103	507	1200	1170.0000

4) Identify customers who have no recorded transactions.

customer_id	first_name	last_name
6	Michael	Wilson
7	Olivia	Martinez
8	William	Taylor
9	Ava	Anderson
10	James	Garcia

5) Calculate the total balance of accounts with no recorded transaction.

total_balance_no_transactions
32300

6) Retrieve transactions for accounts with the lowest balance.

transaction_id	account_id	transaction_type	amount	transaction_date
505	104	withdrawal	800	2023-04-05
508	104	withdrawal	1000	2023-06-10

7) Identify customers who have accounts of multiple types.

```
mysql> SELECT c.customer_id,c.first_name,c.last_name FROM Customers c JOIN (SELECT customer_id FROM Accounts GROUP BY customer_id HAVI
NG COUNT(DISTINCT account_type) > 1) multi_type_accounts ON c.customer_id = multi_type_accounts.customer_id;
Empty set (0.00 sec)
```

8) Calculate the percentage of each account type out of the total number of accounts.

account_type	num_accounts	percentage
savings	4	40.00
current	4	40.00
zero_balance	2	20.00

9) Retrieve all transactions for a customer with a given customer_id.

transaction_type	transaction_date	account_id	account_type	amount
withdrawal	2023-04-05	104	zero_balance	800
withdrawal	2023-06-10	104	zero_balance	1000

10) Calculate the total balance for each account type, including a subquery within the SELECT clause.

account_type	total_balance
savings	33500
current	26850
zero_balance	0

ASSIGNMENT - 4

a Courier Management System with tables for Customers, Couriers, Orders, and Parcels.

TASK -1 DATABASE DESIGN

1) Define the Database Schema • Create SQL tables for entities such as User, Courier, Employee etc.

User

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

Courier

```
mysql> 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
    -> );
```

CourierServices

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

Employee

```
mysql> CREATE TABLE Employee(
    -> EmployeeID INT PRIMARY KEY,
    -> Name VARCHAR(255),
    -> Email VARCHAR(255) UNIQUE,
    -> ContactNumber VARCHAR(20),
    -> Role VARCHAR(50),
    -> Salary DECIMAL(10, 2)
    -> );
```

Location

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

Payment

```
mysql>  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)
    -> );
```

2) Insert sample data into the tables to simulate real-world scenarios.

User

UserID	Name	Email	Password	ContactNumber	Address
1	Aarav Kumar	aarav@gmail.com	pass123	9876543210	12, ABC Colony, New Delhi
2	Sanya Singh	sanya@gmail.com	pass456	8765432109	34, XYZ Street, Mumbai
3	Vivan Patel	vivan@gmail.com	pass789	7654321098	56, PQR Road, Bengaluru
4	Diya Sharma	diya@gmail.com	passabc	6543210987	78, LMN Avenue, Kolkata
5	Advik Gupta	advik@gmail.com	passdef	5432109876	90, EFG Lane, Chennai

Courier

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
101	Rahul Verma	12, ABC Colony, New Delhi	Priya Singh	34, XYZ Street, Mumbai	2.50	In Transit	TRACK001	2023-12-15
102	Neha Sharma	56, PQR Road, Bengaluru	Ajay Kumar	78, LMN Avenue, Kolkata	3.10	Pending	TRACK002	2023-12-18
103	Kriti Patel	90, EFG Lane, Chennai	Arjun Gupta	23, RST Gardens, Hyderabad	1.80	Delivered	TRACK003	2023-12-12
104	Vivek Singh	45, UVW Street, Pune	Shreya Mishra	67, GHI Enclave, Ahmedabad	2.90	In Transit	TRACK004	2023-12-16
105	Ananya Reddy	89, JKL Lane, Jaipur	Rohan Kapoor	10, MNO Road, Chandigarh	2.30	Delivered	TRACK005	2023-12-14

CourierServices

ServiceID	ServiceName	Cost
201	Express Service	299.90
202	Local Delivery	199.99
203	Instant Shipping	149.99
204	Night Delivery	249.99
205	International Shipping	399.99

Employee

EmployeeID	Name	Email	ContactNumber	Role	Salary
301	Rohit Sharma	rohit@gmail.com	9876543210	Manager	70000.00
302	Priya Patel	priya@gmail.com	8765432109	Supervisor	60000.00
303	Anjali Singh	anjali@gmail.com	7654321098	Delivery Driver	50000.00
304	Amit Kumar	amit@gmail.com	6543210987	Customer Service	55000.00
305	Neha Gupta	neha@gmail.com	5432109876	Logistics Coordinator	65000.00

Location

LocationID	LocationName	Address
401	New Delhi	New Delhi, Delhi, India
402	Mumbai	Mumbai, Maharashtra, India
403	Bengaluru	Bengaluru, Karnataka, India
404	Kolkata	Kolkata, West Bengal, India
405	Chennai	Chennai, Tamil Nadu, India

Payment

PaymentID	CourierID	LocationId	Amount	PaymentDate
501	101	401	150.00	2023-12-01
502	102	402	170.00	2023-12-02
503	103	403	245.00	2023-12-03
504	104	404	155.00	2023-12-04
505	105	405	260.00	2023-12-05

TASK-2 SELECT , WHERE

1) List all Customers.

UserID	Name	Email	Password	ContactNumber	Address
1	Aarav Kumar	aarav@gmail.com	pass123	9876543210	12, ABC Colony, New Delhi
2	Sanya Singh	sanya@gmail.com	pass456	8765432109	34, XYZ Street, Mumbai
3	Vivan Patel	vivan@gmail.com	pass789	7654321098	56, PQR Road, Bengaluru
4	Diya Sharma	diya@gmail.com	passabc	6543210987	78, LMN Avenue, Kolkata
5	Advik Gupta	advik@gmail.com	passdef	5432109876	90, EFG Lane, Chennai

2) List all orders for a specific customer:

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
102	Neha Sharma	56, PQR Road, Bengaluru	Ajay Kumar	78, LMN Avenue, Kolkata	3.10	Pending	TRACK002	2023-12-18

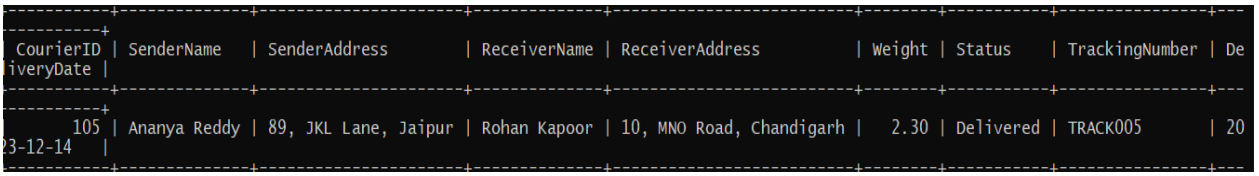
3) List all couriers .

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
101	Rahul Verma	12, ABC Colony, New Delhi	Priya Singh	34, XYZ Street, Mumbai	2.50	In Transit	TRACK001	2023-12-15
102	Neha Sharma	56, PQR Road, Bengaluru	Ajay Kumar	78, LMN Avenue, Kolkata	3.10	Pending	TRACK002	2023-12-18
103	Kriti Patel	90, EFG Lane, Chennai	Arjun Gupta	23, RST Gardens, Hyderabad	1.80	Delivered	TRACK003	2023-12-12
104	Vivek Singh	45, UVW Street, Pune	Shreya Mishra	67, GHI Enclave, Ahmedabad	2.90	In Transit	TRACK004	2023-12-16
105	Ananya Reddy	89, JKL Lane, Jaipur	Rohan Kapoor	10, MNO Road, Chandigarh	2.30	Delivered	TRACK005	2023-12-14

4) List all packages for a specific order:

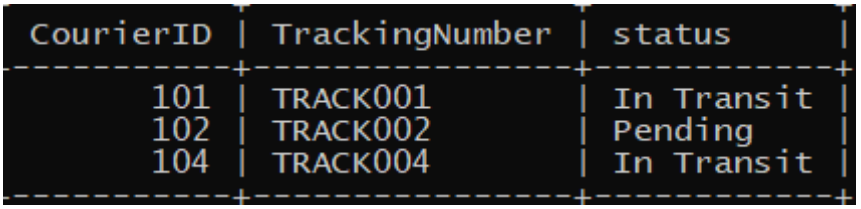
CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
101	Rahul Verma	12, ABC Colony, New Delhi	Priya Singh	34, XYZ Street, Mumbai	2.50	In Transit	TRACK001	2023-12-15

5) . List all deliveries for a specific courier.



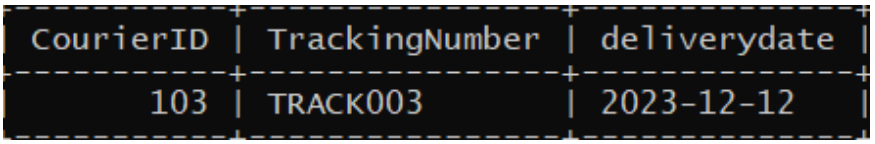
CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
105	Ananya Reddy	89, JKL Lane, Jaipur	Rohan Kapoor	10, MNO Road, Chandigarh	2.30	Delivered	TRACK005	2023-12-14

6) List all undelivered packages.



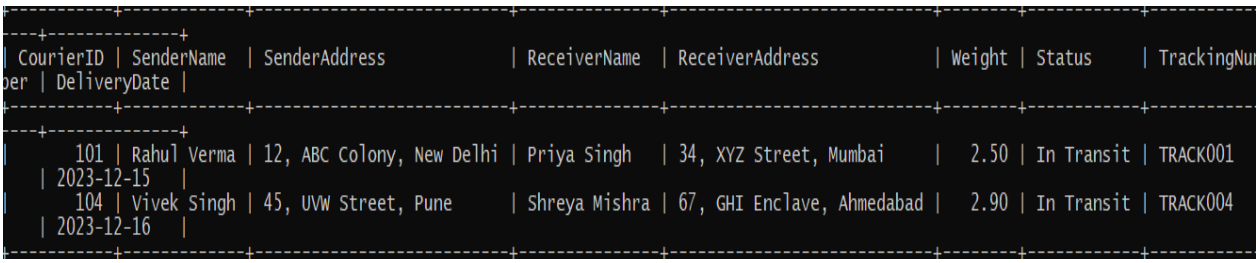
CourierID	TrackingNumber	status
101	TRACK001	In Transit
102	TRACK002	Pending
104	TRACK004	In Transit

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



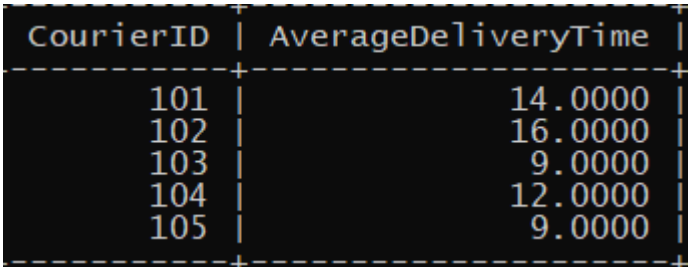
CourierID	TrackingNumber	deliverydate
103	TRACK003	2023-12-12

8) List all packages with a specific status.



CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
101	Rahul Verma	12, ABC Colony, New Delhi	Priya Singh	34, XYZ Street, Mumbai	2.50	In Transit	TRACK001	2023-12-15
104	Vivek Singh	45, UVW Street, Pune	Shreya Mishra	67, GHI Enclave, Ahmedabad	2.90	In Transit	TRACK004	2023-12-16

9) Find the average delivery time for each courier



CourierID	AverageDeliveryTime
101	14.0000
102	16.0000
103	9.0000
104	12.0000
105	9.0000

10) List all packages with a specific weight range.

CourierID	SenderName	ReceiverName	DeliveryDate	weight
105	Ananya Reddy	Rohan Kapoor	2023-12-14	2.30

11) Retrieve employees whose names contain 'John’.

```
ysql> select * from employee where Name like '%John%';
mpty set (0.00 sec)
```

12)Retrieve all courier records with payments greater than $50.

SenderName	ReceiverName	weight	status	TrackingNumber	DeliveryDate	Amount
Rahul Verma	Priya Singh	2.50	In Transit	TRACK001	2023-12-15	150.00
Neha Sharma	Ajay Kumar	3.10	Pending	TRACK002	2023-12-18	170.00
Kriti Patel	Arjun Gupta	1.80	Delivered	TRACK003	2023-12-12	245.00
Vivek Singh	Shreya Mishra	2.90	In Transit	TRACK004	2023-12-16	155.00
Ananya Reddy	Rohan Kapoor	2.30	Delivered	TRACK005	2023-12-14	260.00

TASK 3: GROUPBY, AGGREGATE FUNCTIONS, HAVING, ORDER BY, WHERE

1). Calculate the total revenue generated by each location.

LocationID	LocationName	TotalRevenue
401	New Delhi	150.00
402	Mumbai	170.00
403	Bengaluru	245.00
404	Kolkata	155.00
405	Chennai	260.00

2) Find the total number of couriers delivered to each location.

LocationName	count courier perlocation
New Delhi	1
Mumbai	1
Bengaluru	1
Kolkata	1
Chennai	1

3) Find the courier with the highest average delivery time.

CourierID	AverageDeliveryTime
102	16.0000

4) Find Locations with Total Payments Less Than a Certain Amount .

amount less than 245	locationName
150.00	New Delhi
170.00	Mumbai
155.00	Kolkata

5) Calculate Total Payments per Location.

sum	locationname
150.00	New Delhi
170.00	Mumbai
245.00	Bengaluru
155.00	Kolkata
260.00	Chennai

6) Retrieve couriers who have received payments totaling more than $1000 in a specific location (LocationID = 405).

```
mysql>  select p.Amount , p.courierID  from payment p where p.Amount>1000 and p.locationID  = 405;
Empty set (0.00 sec)
```

7) Retrieve couriers who have received payments totaling more than $150 after a certain date (PaymentDate > '2023-12-02')

Amount	courierID
245.00	103
155.00	104
260.00	105

8) Retrieve locations where the total amount received is more than $200 before a certain date (PaymentDate > ‘2023-12-05’’).

Amount	courierID
150.00	101
170.00	102
155.00	104

TASK 4: INNER JOIN,FULL OUTER JOIN, CROSS JOIN, LEFT OUTER JOIN,RIGHT OUTER JOIN

1) Retrieve Payments with Courier Information .

PaymentID	CourierID	LocationId	Amount	PaymentDate	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
501	101	401	150.00	2023-12-01	101	Rahul Verma	12, ABC Colony, New Delhi	Priya Singh	34, XYZ Street, Mumbai	2.50	In Transit	TRACK001	2023-12-15
502	102	402	170.00	2023-12-02	102	Neha Sharma	56, PQR Road, Bengaluru	Ajay Kumar	78, LMN Avenue, Kolkata	3.10	Pending	TRACK002	2023-12-18
503	103	403	245.00	2023-12-03	103	Kriti Patel	90, EFG Lane, Chennai	Arjun Gupta	23, RST Gardens, Hyderabad	1.80	Delivered	TRACK003	2023-12-12
504	104	404	155.00	2023-12-04	104	Vivek Singh	45, UVW Street, Pune	Shreya Mishra	67, GHI Enclave, Ahmedabad	2.90	In Transit	TRACK004	2023-12-16
505	105	405	260.00	2023-12-05	105	Ananya Reddy	89, JKL Lane, Jaipur	Rohan Kapoor	10, MNO Road, Chandigarh	2.30	Delivered	TRACK005	2023-12-14

2) Retrieve Payments with Location Information.

PaymentID	CourierID	LocationId	Amount	PaymentDate	LocationID	LocationName	Address
501	101	401	150.00	2023-12-01	401	New Delhi	New Delhi, Delhi, India
502	102	402	170.00	2023-12-02	402	Mumbai	Mumbai, Maharashtra, India
503	103	403	245.00	2023-12-03	403	Bengaluru	Bengaluru, Karnataka, India
504	104	404	155.00	2023-12-04	404	Kolkata	Kolkata, West Bengal, India
505	105	405	260.00	2023-12-05	405	Chennai	Chennai, Tamil Nadu, India

3) Retrieve Payments with Courier and Location Information.

amount	CourierId	SenderName	receiverName	LocationName
150.00	101	Rahul Verma	Priya Singh	New Delhi
170.00	102	Neha Sharma	Ajay Kumar	Mumbai
245.00	103	Kriti Patel	Arjun Gupta	Bengaluru
155.00	104	Vivek Singh	Shreya Mishra	Kolkata
260.00	105	Ananya Reddy	Rohan Kapoor	Chennai

4) List all payments with courier details.

Amount	courierID	SenderName	ReceiverName	status	weight
150.00	101	Rahul Verma	Priya Singh	In Transit	2.50
170.00	102	Neha Sharma	Ajay Kumar	Pending	3.10
245.00	103	Kriti Patel	Arjun Gupta	Delivered	1.80
155.00	104	Vivek Singh	Shreya Mishra	In Transit	2.90
260.00	105	Ananya Reddy	Rohan Kapoor	Delivered	2.30

5) Total payments received for each courier.

CourierID	TotalPaymentsReceived
101	150.00
102	170.00
103	245.00
104	155.00
105	260.00

6) List payments made on a specific date.

PaymentID	CourierID	LocationId	Amount	PaymentDate
505	105	405	260.00	2023-12-05

7) Get Courier Information for Each Payment.

payment amount	paymentID	CourierID	SenderName	ReceiverName	status	weight	DeliveryDate
150.00	501	101	Rahul Verma	Priya Singh	In Transit	2.50	2023-12-15
170.00	502	102	Neha Sharma	Ajay Kumar	Pending	3.10	2023-12-18
245.00	503	103	Kriti Patel	Arjun Gupta	Delivered	1.80	2023-12-12
155.00	504	104	Vivek Singh	Shreya Mishra	In Transit	2.90	2023-12-16
260.00	505	105	Ananya Reddy	Rohan Kapoor	Delivered	2.30	2023-12-14

8) Get Payment Details with Location .

PaymentId	amount	paymentDate	LocationName
501	150.00	2023-12-01	New Delhi
502	170.00	2023-12-02	Mumbai
503	245.00	2023-12-03	Bengaluru
504	155.00	2023-12-04	Kolkata
505	260.00	2023-12-05	Chennai

9) Calculating Total Payments for Each Courier.

CourierID	TotalPayments
101	150.00
102	170.00
103	245.00
104	155.00
105	260.00

10) List Payments Within a Date Range .

PaymentID	CourierID	LocationId	Amount	PaymentDate
501	101	401	150.00	2023-12-01
502	102	402	170.00	2023-12-02
503	103	403	245.00	2023-12-03
504	104	404	155.00	2023-12-04

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

```
mysql> select u.name,c.CourierID,c.SenderName,c.ReceiverName,c.status from user u cross join courier c limit 5;
+-------------+-----------+-------------+--------------+------------+
| name        | CourierID | SenderName  | ReceiverName | status     |
+-------------+-----------+-------------+--------------+------------+
| Advik Gupta |       101 | Rahul Verma | Priya Singh  | In Transit |
| Diya Sharma |       101 | Rahul Verma | Priya Singh  | In Transit |
| Vivan Patel |       101 | Rahul Verma | Priya Singh  | In Transit |
| Sanya Singh |       101 | Rahul Verma | Priya Singh  | In Transit |
| Aarav Kumar |       101 | Rahul Verma | Priya Singh  | In Transit |
+-------------+-----------+-------------+--------------+------------+
5 rows in set (0.00 sec)
```

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

```
mysql> SELECT DISTINCT cs.ServiceName , c.SenderName , c.Receivername FROM Courier c CROSS JOIN CourierServices cs limit 5;
+----------------+-------------+--------------+
| ServiceName    | SenderName  | Receivername |
+----------------+-------------+--------------+
| Express Service | Ananya Reddy | Rohan Kapoor |
| Express Service | Vivek Singh  | Shreya Mishra |
| Express Service | Kriti Patel  | Arjun Gupta  |
| Express Service | Neha Sharma  | Ajay Kumar   |
| Express Service | Rahul Verma  | Priya Singh  |
+----------------+-------------+--------------+
```

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

EmployeeID	Name	paymentID	Amount
301	Rohit Sharma	505	260.00
301	Rohit Sharma	504	155.00
301	Rohit Sharma	503	245.00
301	Rohit Sharma	502	170.00
301	Rohit Sharma	501	150.00

14) List all users and all courier services, showing all possible combinations.

UserID	Name	Email	Password	ContactNumber	Address	ServiceID	ServiceName	Cost
5	Advik Gupta	advik@gmail.com	passdef	5432109876	90, EFG Lane, Chennai	201	Express Service	299.90
4	Diya Sharma	diya@gmail.com	passabc	6543210987	78, LMN Avenue, Kolkata	201	Express Service	299.90
3	Vivan Patel	vivan@gmail.com	pass789	7654321098	56, PQR Road, Bengaluru	201	Express Service	299.90
2	Sanya Singh	sanya@gmail.com	pass456	8765432109	34, XYZ Street, Mumbai	201	Express Service	299.90
1	Aarav Kumar	aarav@gmail.com	pass123	9876543210	12, ABC Colony, New Delhi	201	Express Service	299.90

15) List all employees and all locations, showing all possible combinations:

EmployeeID	Name	Email	ContactNumber	Role	Salary	LocationID	LocationName	Address
305	Neha Gupta	neha@gmail.com	5432109876	Logistics Coordinator	65000.00	401	New Delhi	New Delhi, Delhi, India
304	Amit Kumar	amit@gmail.com	6543210987	Customer Service	55000.00	401	New Delhi	New Delhi, Delhi, India
303	Anjali Singh	anjali@gmail.com	7654321098	Delivery Driver	50000.00	401	New Delhi	New Delhi, Delhi, India
302	Priya Patel	priya@gmail.com	8765432109	Supervisor	60000.00	401	New Delhi	New Delhi, Delhi, India
301	Rohit Sharma	rohit@gmail.com	9876543210	Manager	70000.00	401	New Delhi	New Delhi, Delhi, India

16) Retrieve a list of couriers and their corresponding sender information.

courierID	SenderName	SenderAddress
101	Rahul Verma	12, ABC Colony, New Delhi
102	Neha Sharma	56, PQR Road, Bengaluru
103	Kriti Patel	90, EFG Lane, Chennai
104	Vivek Singh	45, UVW Street, Pune
105	Ananya Reddy	89, JKL Lane, Jaipur

17) Retrieve a list of couriers and their corresponding receiver information.

CourierID	ReceiverName	ReceiverAddress
101	Priya Singh	34, XYZ Street, Mumbai
102	Ajay Kumar	78, LMN Avenue, Kolkata
103	Arjun Gupta	23, RST Gardens, Hyderabad
104	Shreya Mishra	67, GHI Enclave, Ahmedabad
105	Rohan Kapoor	10, MNO Road, Chandigarh

18) Retrieve a list of couriers along with the courier service details.

courierID	ServiceID	ServiceName
105	201	Express Service
104	201	Express Service
103	201	Express Service
102	201	Express Service
101	201	Express Service

19) Retrieve a list of locations and the total payment amount received at each location.

Amount	locationName
150.00	New Delhi
170.00	Mumbai
245.00	Bengaluru
155.00	Kolkata
260.00	Chennai

20) Retrieve all couriers sent by the same sender.

```
mysql> select courierID , SenderName ,count(courierID) from courier group  by SenderName,courierID having count(*)>1;
Empty set (0.00 sec)
```

21) List all employees who share the same role.

EmployeeID	Name	Email	ContactNumber	Role	Salary
302	Priya Patel	priya@gmail.com	8765432109	Delivery Driver	60000.00
303	Anjali Singh	anjali@gmail.com	7654321098	Delivery Driver	50000.00

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

payment amount	Address
150.00	New Delhi , Delhi,India
260.00	New Delhi , Delhi,India

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

courierID	Weight
102	3.10
104	2.90

24) Find the names of all employees who have a salary greater than the average salary.

Name
Rohit Sharma
Neha Gupta

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

TotalCost
899.87

26) Find all couriers that have been paid for.

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
101	Rahul Verma	12, ABC Colony, New Delhi	Priya Singh	34, XYZ Street, Mumbai	2.50	In Transit	TRACK001	2023-12-15
102	Neha Sharma	56, PQR Road, Bengaluru	Ajay Kumar	78, LMN Avenue, Kolkata	3.10	Pending	TRACK002	2023-12-18
103	Kriti Patel	90, EFG Lane, Chennai	Arjun Gupta	23, RST Gardens, Hyderabad	1.80	Delivered	TRACK003	2023-12-12
104	Vivek Singh	45, UVW Street, Pune	Shreya Mishra	67, GHI Enclave, Ahmedabad	2.90	In Transit	TRACK004	2023-12-16
105	Ananya Reddy	89, JKL Lane, Jaipur	Rohan Kapoor	10, MNO Road, Chandigarh	2.30	Delivered	TRACK005	2023-12-14

27) Find the locations where the maximum payment amount was made .

LocationName
Chennai

28) Find all couriers whose weight is greater than the weight of all couriers sent by a specific sender.

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
102	Neha Sharma	56, PQR Road, Bengaluru	Ajay Kumar	78, LMN Avenue, Kolkata	3.10	Pending	TRACK002	2023-12-18
104	Vivek Singh	45, UVW Street, Pune	Shreya Mishra	67, GHI Enclave, Ahmedabad	2.90	In Transit	TRACK004	2023-12-16

ASSIGNMENT - 5

This assignment task is to create a ticket booking system for an Event. The system should support booking tickets for different types of events, such as movies, concerts, and plays. Each event has its own pricing strategy, and the system should also track available seats and customer bookings

TASK -1 DATABASE DESIGN

1) Create a database named "TicketBookingSystem".

```
mysql> CREATE DATABASE TicketBookingSystem;
Query OK, 1 row affected (0.03 sec)
```

2) Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

Venu

```
mysql> CREATE TABLE VENU(
    ->  venue_id INT PRIMARY KEY ,
    -> venue_name VARCHAR(255),
    ->  address TEXT
    -> );
```

Event

```
mysql> CREATE TABLE Event(
    -> event_id INT PRIMARY KEY,
    -> event_name VARCHAR(50),
    -> event_time TIME,
    ->  venue_id INT,
    ->  total_seats INT,
    ->  available_seats INT ,
    ->  ticket_price DECIMAL(4,2),
    ->  event_type ENUM('Movie','Sports','Concert'),
    ->  booking_id INT,
    ->  FOREIGN KEY(venue_id) REFERENCES VENU(venue_id)
    -> );
```

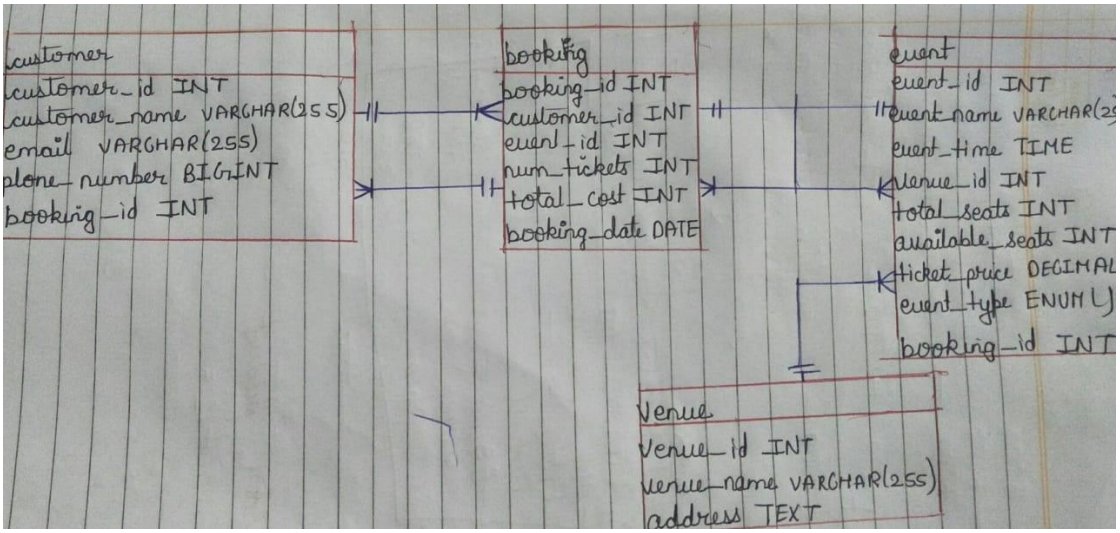
Customers

```
mysql> CREATE TABLE Customer(
    -> customer_id INT PRIMARY KEY,
    -> customer_name VARCHAR(255),
    -> email VARCHAR(255),
    -> phone_number BIGINT,
    -> booking_id INT
    -> );
```

Booking

```
mysql> CREATE TABLE BOOKING(
    -> booking_id INT PRIMARY KEY,
    -> customer_id INT,
    -> event_id INT,
    -> num_tickets INT,
    -> total_cost INT,
    -> booking_date DATE,
    ->  FOREIGN KEY(customer_id)  REFERENCES Customer(customer_id),
    -> FOREIGN KEY(event_id) REFERENCES Event(event_id)
    -> );
```

3) Create an ERD (Entity Relationship Diagram) for the database.



4) Create appropriate Primary Key and Foreign Key constraints for referential integrity.

Primary Key

```
+------------+-----------------+
| TABLE_NAME | CONSTRAINT_NAME |
+------------+-----------------+
| customer   | PRIMARY         |
| venu       | PRIMARY         |
| event      | PRIMARY         |
+------------+-----------------+
```

Foreign Key

TABLE_NAME	CONSTRAINT_NAME
customer	fk_customer_address
customer	fk_customer_store
event	event_ibfk_1
event	event_ibfk_2
customer	customer_ibfk_1

TASKS_2: SELECT, WHERE, BETWEEN, AND, LIKE

1) Write a SQL query to insert at least 10 sample records into each table.

Venu

venue_id	venue_name	address
101	Rajasthan Convention Center	1 Jaipur Road, Delhi
102	Taj Mahal Auditorium	22 Agra Street, Mumbai
103	Bollywood Studios	33 Film City, Mumbai
104	Ganges Stadium	44 Varanasi Avenue, Kolkata
105	Lotus Palace Theater	55 Bangalore Circle, Bangalore
106	Chennai Music Hall	66 Carnatic Street, Chennai
107	Punjab Sports Arena	77 Amritsar Road, Chandigarh
108	Kerala Cultural Center	88 Cochin Lane, Kochi
109	Hyderabad Exhibition Center	99 Hitech City, Hyderabad
110	Gujarat Pavilion	100 Ahmedabad Street, Gandhinagar

Event

event_id	event_name	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id	event_date
201	Movie Night	18:00:00	101	200	200	10.00	Movie	NULL	2023-12-20
202	Concert Extravaganza	20:30:00	102	500	500	50.00	Concert	NULL	2024-01-15
203	Sports Championship	15:00:00	103	1000	800	20.00	Sports	NULL	2024-02-05
204	Drama Play	19:30:00	104	150	100	15.00	Movie	NULL	2024-03-10
205	Live Comedy Show	21:00:00	105	300	300	25.00	Concert	NULL	2024-04-18
206	Musical Night	19:00:00	106	250	200	30.00	Concert	NULL	2024-05-25
207	Dance Performance	17:30:00	107	400	350	18.00	Movie	NULL	2024-06-30
208	Art Exhibition	12:00:00	108	120	100	12.00	Sports	NULL	2024-07-12
209	Magic Show	16:00:00	109	180	150	8.00	Sports	NULL	2024-08-20
210	Cultural Festival	14:00:00	110	600	550	22.00	Concert	NULL	2024-09-28

Customer

customer_id	customer_name	email	phone_number	booking_id
301	Aarav Patel	aarav@example.com	876543210	1
302	Aadhya Singh	aadhya@example.com	876532109	2
303	Advik Gupta	advik@example.com	7654321098	3
304	Ananya Reddy	ananya@example.com	6543210987	4
305	Ishaan Kumar	ishaan@example.com	9432109876	5
306	Aaradhya Joshi	aaradhya@example.com	4321098765	6
307	Vihaan Khanna	vihaan@example.com	9210987654	7
308	Kyra Sharma	kyra@example.com	92109876543	8
309	Reyansh Verma	reyansh@example.com	8098765432	9
310	Saisha Mishra	saisha@example.com	70987654321	10

Booking

booking_id	customer_id	event_id	num_tickets	total_cost	booking_date
1	301	201	2	420	2023-12-18
2	302	202	3	150	2024-01-10
3	303	203	5	100	2024-02-01
4	304	204	1	215	2024-03-05
5	305	205	4	100	2024-04-12
6	306	206	2	660	2024-05-20
7	307	207	3	454	2024-06-25
8	308	208	2	124	2024-07-05
9	309	209	4	932	2024-08-15
10	310	210	6	132	2024-09-22

2) Write a SQL query to list all Events.

event_id	event_name	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id	event_date
201	Movie Night	18:00:00	101	200	200	10.00	Movie	NULL	2023-12-20
202	Concert Extravaganza	20:30:00	102	500	500	50.00	Concert	NULL	2024-01-15
203	Sports Championship	15:00:00	103	1000	800	20.00	Sports	NULL	2024-02-05
204	Drama Play	19:30:00	104	150	100	15.00	Movie	NULL	2024-03-10
205	Live Comedy Show	21:00:00	105	300	300	25.00	Concert	NULL	2024-04-18
206	Musical Night	19:00:00	106	250	200	30.00	Concert	NULL	2024-05-25
207	Dance Performance	17:30:00	107	400	350	18.00	Movie	NULL	2024-06-30
208	Art Exhibition	12:00:00	108	120	100	12.00	Sports	NULL	2024-07-12
209	Magic Show	16:00:00	109	180	150	8.00	Sports	NULL	2024-08-20
210	Cultural Festival	14:00:00	110	600	550	22.00	Concert	NULL	2024-09-28

3) Write a SQL query to select events with available tickets

event_id	event_name	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id	event_date
201	Movie Night	18:00:00	101	200	200	10.00	Movie	NULL	2023-12-20
202	Concert Extravaganza	20:30:00	102	500	500	50.00	Concert	NULL	2024-01-15
203	Sports Championship	15:00:00	103	1000	800	20.00	Sports	NULL	2024-02-05
204	Drama Play	19:30:00	104	150	100	15.00	Movie	NULL	2024-03-10
205	Live Comedy Show	21:00:00	105	300	300	25.00	Concert	NULL	2024-04-18
206	Musical Night	19:00:00	106	250	200	30.00	Concert	NULL	2024-05-25
207	Dance Performance	17:30:00	107	400	350	18.00	Movie	NULL	2024-06-30
208	Art Exhibition	12:00:00	108	120	100	12.00	Sports	NULL	2024-07-12
209	Magic Show	16:00:00	109	180	150	8.00	Sports	NULL	2024-08-20
210	Cultural Festival	14:00:00	110	600	550	22.00	Concert	NULL	2024-09-28

4) Write a SQL query to select events name partial match with ‘cup’.

```
mysql> SELECT *
    -> FROM Event
    -> WHERE event_name LIKE '%cup%';
Empty set (0.00 sec)
```

5) Write a SQL query to select events with ticket price range between 1000 to 2500.

```
mysql> SELECT *
    -> FROM Event
    -> WHERE ticket_price BETWEEN 1000 AND 2500;
Empty set (0.00 sec)
```

6) Write a SQL query to retrieve events with dates falling within a specific range.

event_id	event_name	event_time	venue_id	ticket_price	event_type
201	Movie Night	18:00:00	101	10.00	Movie

7) Write a SQL query to retrieve events with available tickets that also have "Concert" in their name.

event_id	event_name	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id	event_date
202	Concert Extravaganza	20:30:00	102	500	500	50.00	Concert	NULL	2024-01-15

8) Write a SQL query to retrieve users in batches of 5, starting from the 6th user.

customer_id	customer_name	email	phone_number	booking_id
306	Aaradhya Joshi	aaradhya@example.com	4321098765	6
307	Vihaan Khanna	vihaan@example.com	9210987654	7
308	Kyra Sharma	kyra@example.com	92109876543	8
309	Reyansh Verma	reyansh@example.com	8098765432	9
310	Saisha Mishra	saisha@example.com	70987654321	10

9) Write a SQL query to retrieve bookings details containing booked no of ticket more than 4 .

booking_id	customer_id	event_id	num_tickets	total_cost	booking_date
3	303	203	5	100	2024-02-01
10	310	210	6	132	2024-09-22

10) Write a SQL query to retrieve customer information whose phone number end with '000'.

```
ysql> use ticketbookingsystem;
atabase changed
ysql> SELECT *
   -> FROM Customer
   -> WHERE phone_number LIKE '%000';
mpty set (0.00 sec)
```

11) Write a SQL query to retrieve the events in order whose seat capacity more than 15000.

```
mysql> SELECT *
    -> FROM Event
    -> WHERE total_seats > 15000
    -> ORDER BY total_seats ASC;
Empty set (0.00 sec)

mysql>
```

12) Write a SQL query to select events name not start with 'x', 'y', 'z'.

event_id	event_name	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id	event_date
201	Movie Night	18:00:00	101	200	200	10.00	Movie	NULL	2023-12-20
202	Concert Extravaganza	20:30:00	102	500	500	50.00	Concert	NULL	2024-01-15
203	Sports Championship	15:00:00	103	1000	800	20.00	Sports	NULL	2024-02-05
204	Drama Play	19:30:00	104	150	100	15.00	Movie	NULL	2024-03-10
205	Live Comedy Show	21:00:00	105	300	300	25.00	Concert	NULL	2024-04-18
206	Musical Night	19:00:00	106	250	200	30.00	Concert	NULL	2024-05-25
207	Dance Performance	17:30:00	107	400	350	18.00	Movie	NULL	2024-06-30
208	Art Exhibition	12:00:00	108	120	100	12.00	Sports	NULL	2024-07-12
209	Magic Show	16:00:00	109	180	150	8.00	Sports	NULL	2024-08-20
210	Cultural Festival	14:00:00	110	600	550	22.00	Concert	NULL	2024-09-28

TASK 3: AGGREGATE FUNCTION, HAVING, ORDER BY, GROUPBY and JOINS

1) Write a SQL query to List Events and Their Average Ticket Prices.

event_id	event_name	average_ticket_price
201	Movie Night	10.000000
202	Concert Extravaganza	50.000000
203	Sports Championship	20.000000
204	Drama Play	15.000000
205	Live Comedy Show	25.000000
206	Musical Night	30.000000
207	Dance Performance	18.000000
208	Art Exhibition	12.000000
209	Magic Show	8.000000
210	Cultural Festival	22.000000

2) Write a SQL query to Calculate the Total Revenue Generated by Events.

total_revenue
687.00

3) Write a SQL query to find the event with the highest ticket sales.

event_id	event_name	total_tickets_sold
210	Cultural Festival	6

4) Write a SQL query to Find Events with No Ticket Sales.

```
mysql> SELECT E.event_id, E.event_name
    -> FROM Event E
    -> LEFT JOIN Booking B ON E.event_id = B.event_id
    -> WHERE B.booking_id IS NULL;
Empty set (0.00 sec)
```

5) Write a SQL query to Find the User Who Has Booked the Most Tickets.

customer_id	total_tickets_booked
310	6

6) Write a SQL query to Calculate the Total Number of Tickets Sold for Each Event.

event_id	total_tickets_sold
201	2
202	3
203	5
204	1
205	4
206	2
207	3
208	2
209	4
210	6

7) Write a SQL query to List Events and the total number of tickets sold for each month.

event_id	event_name	month	total_tickets_sold
202	Concert Extravaganza	1	3
203	Sports Championship	2	5
204	Drama Play	3	1
205	Live Comedy Show	4	4
206	Musical Night	5	2
207	Dance Performance	6	3
208	Art Exhibition	7	2
209	Magic Show	8	4
210	Cultural Festival	9	6
201	Movie Night	12	2

8) Write a SQL query to calculate the average Ticket Price for Events in Each Venue.

venue_id	venue_name	average_ticket_price
101	Rajasthan Convention Center	10.000000
102	Taj Mahal Auditorium	50.000000
103	Bollywood Studios	20.000000
104	Ganges Stadium	15.000000
105	Lotus Palace Theater	25.000000
106	Chennai Music Hall	30.000000
107	Punjab Sports Arena	18.000000
108	Kerala Cultural Center	12.000000
109	Hyderabad Exhibition Center	8.000000
110	Gujarat Pavilion	22.000000

9) Write a SQL query to calculate the total Revenue Generated by Events in Each Year.

year	total_revenue
2023	20.00
2024	667.00

10) Write a SQL query to list users who have booked tickets for multiple events.

```
mysql> SELECT customer_id, COUNT(DISTINCT event_id) AS num_events_booked
    -> FROM Booking
    -> GROUP BY customer_id
    -> HAVING COUNT(DISTINCT event_id) > 1;
Empty set (0.00 sec)
```

11) Write a SQL query to calculate the Total Revenue Generated by Events for Each User.

customer_id	total_revenue
301	20.00
302	150.00
303	100.00
304	15.00
305	100.00
306	60.00
307	54.00
308	24.00
309	32.00
310	132.00

12) Write a SQL query to calculate the Average Ticket Price for Events in Each Category and Venue.

event_type	venue_id	average_ticket_price
Movie	101	10.000000
Concert	102	50.000000
Sports	103	20.000000
Movie	104	15.000000
Concert	105	25.000000
Concert	106	30.000000
Movie	107	18.000000
Sports	108	12.000000
Sports	109	8.000000
Concert	110	22.000000

13) Write a SQL query to list Users and the Total Number of Tickets They've Purchased in the Last 30 Days.

customer_id	total_tickets_purchased
301	2
302	3
303	5
304	1
305	4
306	2
307	3
308	2
309	4
310	6

TASKS 4: SUBQUERY and IT'S TYPES

1) Calculate the Average Ticket Price for Events in Each Venue Using a Subquery.

venue_id	venue_name	average_ticket_price_per_venue
101	Rajasthan Convention Center	10.000000
102	Taj Mahal Auditorium	50.000000
103	Bollywood Studios	20.000000
104	Ganges Stadium	15.000000
105	Lotus Palace Theater	25.000000
106	Chennai Music Hall	30.000000
107	Punjab Sports Arena	18.000000
108	Kerala Cultural Center	12.000000
109	Hyderabad Exhibition Center	8.000000
110	Gujarat Pavilion	22.000000

2) Find Events with More Than 50% of Tickets Sold using subquery.

```
mysql> SELECT *
    -> FROM Event
    -> WHERE (
    ->     SELECT SUM(num_tickets)
    ->     FROM Booking
    ->     WHERE Booking.event_id = Event.event_id
    -> ) > (0.5 * total_seats);
Empty set (0.00 sec)
```

3) Calculate the Total Number of Tickets Sold for Each Event.

event_id	total_tickets_sold
201	2
202	3
203	5
204	1
205	4
206	2
207	3
208	2
209	4
210	6

4) Find Users Who Have Not Booked Any Tickets Using a NOT EXISTS Subquery.

```
mysql> SELECT *
    -> FROM Customer C
    -> WHERE NOT EXISTS (
    ->     SELECT 1
    ->     FROM Booking B
    ->     WHERE B.customer_id = C.customer_id
    -> );
Empty set (0.00 sec)
```

5) List Events with No Ticket Sales Using a NOT IN Subquery.

```
mysql> SELECT *
    -> FROM Event
    -> WHERE event_id NOT IN (
    ->     SELECT DISTINCT event_id
    ->     FROM Booking
    -> );
Empty set (0.00 sec)
```

6) Calculate the Total Number of Tickets Sold for Each Event Type Using a Subquery in the FROM Clause.

event_type	total_tickets_sold
Movie	6
Concert	15
Sports	11

7) Find Events with Ticket Prices Higher Than the Average Ticket Price Using a Subquery in the WHERE Clause.

event_id	event_name	event_time	venue_id	total_seats	available_seats	ticket_price	event_type	booking_id	event_date
202	Concert Extravaganza	20:30:00	102	500	500	50.00	Concert	NULL	2024-01-15
205	Live Comedy Show	21:00:00	105	300	300	25.00	Concert	NULL	2024-04-18
206	Musical Night	19:00:00	106	250	200	30.00	Concert	NULL	2024-05-25
210	Cultural Festival	14:00:00	110	600	550	22.00	Concert	NULL	2024-09-28

8) Calculate the Total Revenue Generated by Events for Each User Using a Correlated Subquery

customer_id	total_revenue
301	20.00
302	150.00
303	100.00
304	15.00
305	100.00
306	60.00
307	54.00
308	24.00
309	32.00
310	132.00

9) List Users Who Have Booked Tickets for Events in a Given Venue Using a Subquery in the WHERE Clause.

customer_id	customer_name	email	phone_number	booking_id
306	Aaradhya Joshi	aaradhya@example.com	4321098765	6

10) Calculate the Total Number of Tickets Sold for Each Event Category Using a Subquery with GROUP BY.

event_type	total_tickets_sold_per_category
Movie	6
Concert	15
Sports	11

11) Find Users Who Have Booked Tickets for Events in each Month Using a Subquery with DATE_FORMAT.

customer_id	booked_month
301	12
302	1
303	2
304	3
305	4
306	5
307	6
308	7
309	8
310	9

12) Calculate the Average Ticket Price for Events in Each Venue Using a Subquery.

venue_id	venue_name	average_ticket_price_per_venue
101	Rajasthan Convention Center	10.000000
102	Taj Mahal Auditorium	50.000000
103	Bollywood Studios	20.000000
104	Ganges Stadium	15.000000
105	Lotus Palace Theater	25.000000
106	Chennai Music Hall	30.000000
107	Punjab Sports Arena	18.000000
108	Kerala Cultural Center	12.000000
109	Hyderabad Exhibition Center	8.000000
110	Gujarat Pavilion	22.000000