

SQL – Hands on assignment

A) Library

Q1) Create a table BOOKS with the given schema.

-> CREATE DATABASE library;

USE library;

```
CREATE TABLE books(  
    book_id INT PRIMARY KEY,  
    title VARCHAR(45),  
    author VARCHAR(45),  
    genre VARCHAR(45),  
    price INT,  
    publication_year INT,  
    copies INT  
);
```

-> Output:

Field	Type	Null	Key	Default	Extra
book_id	int	NO	PRI	NULL	
title	varchar(45)	YES		NULL	
author	varchar(45)	YES		NULL	
genre	varchar(45)	YES		NULL	
price	int	YES		NULL	
publication_year	int	YES		NULL	
copies	int	YES		NULL	

Q2) Insert at least 5 rows into the BOOKS table.

```
-> INSERT INTO books VALUES(1, "David Copperfield", "Pratyush", "Mystery", 200, 2010, 10);
```

```
INSERT INTO books VALUES(2, "Poseidon", "Ruskin Bond", "Fiction", 600, 2012, 0);
```

```
INSERT INTO books VALUES(3, "Sherlock Holmes", "Christina", "Mystery", 400, 2018, 18);
```

```
INSERT INTO books VALUES(4, "Vaidya", "Rabindranath Tagore", "Healthcare", 150, 2016, 40);
```

```
INSERT INTO books VALUES(5, "One Piece", "Oda", "Adventure", 700, 2004, 50);
```

Q3) Display all the details of books available in the library.

-> SELECT * FROM books;

-> Output:

book_id	title	author	genre	price	publication_year	copies
1	David Copperfield	Pratyush	Mystery	200	2010	10
2	Poseidon	Ruskin Bond	Fiction	600	2012	0
3	Sherlock Holmes	Christina	Mystery	400	2018	18
4	Vaidya	Rabindranath Tagore	Healthcare	150	2016	40
5	One Piece	Oda	Adventure	700	2004	50

Q4) Display the list of books published after 2015.

-> SELECT * FROM books WHERE publication_year > 2015;

-> Output:

book_id	title	author	genre	price	publication_year	copies
3	Sherlock Holmes	Christina	Mystery	400	2018	18
4	Vaidya	Rabindranath Tagore	Healthcare	150	2016	40

Q5) Create a table BORROWERS with the given schema.

```
-> CREATE TABLE borrowers(  
    borrower_id INT PRIMARY KEY,  
    name VARCHAR(45),  
    address VARCHAR(60),  
    phone VARCHAR(10),  
    membership_type VARCHAR(30)  
);
```

-> Output:

Field	Type	Null	Key	Default	Extra
borrower_id	int	NO	PRI	<div>NONE</div>	
name	varchar(45)	YES		<div>NONE</div>	
address	varchar(60)	YES		<div>NONE</div>	
phone	varchar(10)	YES		<div>NONE</div>	
membership_type	varchar(30)	YES		<div>NONE</div>	

Q6) Insert at least 5 rows into the BORROWERS table.

```
-> INSERT INTO borrowers VALUES(1, "Shahid", "Kothrud", "7666234983", "Gold");  
INSERT INTO borrowers VALUES(2, "Prayushi", "Nigdi", "7050576230", "Silver");  
INSERT INTO borrowers VALUES(3, "Sneha", "Andheri", "9922449438", "Bronze");  
INSERT INTO borrowers VALUES(4, "Disha", "Katraj", "9545447745", "Gold");  
INSERT INTO borrowers VALUES(5, "Karthik", "Pimpri", "7009124589", "Silver");
```

Q7) Display the names and phone numbers of all borrowers.

-> SELECT name, phone FROM borrowers;

-> Output:

name	phone
Shahid	7666234983
Prayushi	7050576230
Sneha	9922449438
Disha	9545447745
Karthik	7009124589

Q8) Display the list of borrowers who have a "Gold" membership type.

-> SELECT * FROM borrowers WHERE membership_type = "Gold";

-> Output:

borrower_id	name	address	phone	membership_type
1	Shahid	Kothrud	7666234983	Gold
4	Disha	Katraj	9545447745	Gold

Q9) Create a table ISSUES with the given schema.

```
-> CREATE TABLE issues(  
    issue_id INT PRIMARY KEY,  
    borrower_id INT,  
    book_id INT,  
    issue_date DATE,  
    return_date DATE  
);
```

-> Output:

Field	Type	Null	Key	Default	Extra
issue_id	int	NO	PRI	NULL	
borrower_id	int	YES		NULL	
book_id	int	YES		NULL	
issue_date	date	YES		NULL	
return_date	date	YES		NULL	

Q10) Insert 5 records into the ISSUES table.

```
-> INSERT INTO issues(issue_id, borrower_id, book_id, issue_date) VALUES(1, 1, 1, "2016-03-01");
```

```
INSERT INTO issues VALUES(2, 1, 3, "2018-03-01", "2018-04-01");
```

```
INSERT INTO issues VALUES(3, 1, 5, "2017-08-10", "2017-09-18");
```

```
INSERT INTO issues VALUES(4, 3, 4, "2014-02-10", "2014-05-30");
```

```
INSERT INTO issues VALUES(5, 2, 1, "2019-11-07", "2019-12-30");
```

Q11) Display the title and author of all books priced above 500.

-> SELECT title, author FROM books WHERE price > 500;

-> Output:

title	author
Poseidon	Ruskin Bond
One Piece	Oda

Q12) Update the price of all books in the "Fiction" genre by increasing it by 10%.

-> UPDATE books SET price = price + (price/10) WHERE genre = "Fiction";

-> Output:

book_id	title	author	genre	price	publication_year	copies
2	Poseidon	Ruskin Bond	Fiction	660	2012	0

Q13) Delete the records of books that have no copies left.

-> DELETE FROM books WHERE copies = 0;

-> Output:

book_id	title	author	genre	price	publication_year	copies
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Q14) Create a view AVAILABLE_BOOKS showing all books with more than 5 copies.

-> CREATE VIEW AVAILABLE_BOOKS AS

SELECT copies FROM books WHERE copies > 5;

-> Output:

copies
10
18
40
50

Q15) Retrieve all the books sorted by Publication_Year in descending order.

-> SELECT * FROM books ORDER BY publication_year DESC;

-> Output:

book_id	title	author	genre	price	publication_year	copies
3	Sherlock Holmes	Christina	Mystery	400	2018	18
4	Vaidya	Rabindranath Tagore	Healthcare	150	2016	40
1	David Copperfield	Pratyush	Mystery	200	2010	10
5	One Piece	Oda	Adventure	700	2004	50

Q16) Retrieve the details of borrowers who borrowed more than 2 books.

-> SELECT b.borrower_id, b.name, b.address, b.phone, b.membership_type

FROM borrowers b

JOIN issues u

ON b.borrower_id = u.borrower_id

GROUP BY u.borrower_id

HAVING COUNT(u.borrower_id) > 2;

-> Output:

borrower_id	name	address	phone	membership_type
1	Shahid	Kothrud	7666234983	Gold

Q17) Display the name of borrower who borrowed book but never returned.

-> SELECT b.name

FROM borrowers b

JOIN issues u

ON b.borrower_id = u.borrower_id

WHERE u.return_date IS NULL;

-> Output:

name
Shahid

B) Movie

Q1) Create a table MOVIES with the given schema.

-> CREATE DATABASE movie;

USE movie;

```
CREATE TABLE movies(  
    movie_id INT PRIMARY KEY,  
    title VARCHAR(45),  
    genre VARCHAR(45),  
    release_date DATE,  
    rating INT,  
    director VARCHAR(45)  
);
```

-> Output:

Field	Type	Null	Key	Default	Extra
movie_id	int	NO	PRI	NULL	
title	varchar(45)	YES		NULL	
genre	varchar(45)	YES		NULL	
release_date	date	YES		NULL	
rating	int	YES		NULL	
director	varchar(45)	YES		NULL	

Q2) Insert at least 5 rows into the MOVIES table.

```
-> INSERT INTO movies VALUES(1, "Action Jackson", "Action", "2010-05-01", 9, "Rohit Shetty");
```

```
INSERT INTO movies VALUES(2, "Cyberpunk", "Sci-Fi", "2023-06-18", 8, "James Coley");
```

```
INSERT INTO movies VALUES(3, "One Piece Film Red", "Adventure", "2024-02-01", 10, "Oda Eichiro");
```

```
INSERT INTO movies VALUES(4, "Kabhi Khushi Kabhi Gham", "Romance", "2004-07-11", 10, "Karan Johar");
```

```
INSERT INTO movies VALUES(5, "Boss", "Action", "2016-12-01", 4, "Rohit Shetty");
```

Q3) Display all the details of movies available.

-> SELECT * FROM movies;

-> Output:

movie_id	title	genre	release_date	rating	director
1	Action Jackson	Action	2010-05-01	9	Rohit Shetty
2	Cyberpunk	Sci-Fi	2023-06-18	8	James Coley
3	One Piece Film Red	Adventure	2024-02-01	10	Oda Eichiro
4	Kabhi Khushi Kabhi Gham	Romance	2004-07-11	10	Karan Johar
5	Boss	Action	2016-12-01	4	Rohit Shetty

Q4) Display the list of movies in the "Action" genre.

-> SELECT * FROM movies WHERE genre = "Action";

-> Output:

movie_id	title	genre	release_date	rating	director
1	Action Jackson	Action	2010-05-01	9	Rohit Shetty
5	Boss	Action	2016-12-01	4	Rohit Shetty

Q5) Create a table CUSTOMERS with the given schema.

```
-> CREATE TABLE customer(  
    customer_id INT PRIMARY KEY,  
    name VARCHAR(45),  
    email VARCHAR(45),  
    phone VARCHAR(10),  
    membership_type VARCHAR(45)  
);
```

-> Output:

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	<div>NULL</div>	
name	varchar(45)	YES		<div>NULL</div>	
email	varchar(45)	YES		<div>NULL</div>	
phone	varchar(10)	YES		<div>NULL</div>	
membership_type	varchar(45)	YES		<div>NULL</div>	

Q6) Insert at least 5 rows into the CUSTOMERS table.

```
-> INSERT INTO customer VALUES(1, "Pratyush", "pratyushm@gmail.com",  
"7666234983", "Gold");
```

```
INSERT INTO customer VALUES(2, "Prakash", "prakashm@gmail.com",  
"9922449438", "Premium");
```

```
INSERT INTO customer VALUES(3, "Suchismita", "suchismitam@gmail.com",  
"9545447745", "Gold");
```

```
INSERT INTO customer VALUES(4, "Prayushi", "prayushim@gmail.com",  
"7057521969", "Silver");
```

```
INSERT INTO customer(customer_id, name, email, phone) VALUES(5, "Pankaj",  
"pankajm@gmail.com", "7662214980");
```

Q7) Display the names and emails of all customers.

-> SELECT name, email FROM customer;

-> Output:

name	email
Pratyush	pratyushm@gmail.com
Prakash	prakashm@gmail.com
Suchismita	suchismitam@gmail.com
Prayushi	prayushim@gmail.com
Pankaj	pankajm@gmail.com

Q8) Display the list of customers with a "Premium" membership.

-> SELECT * FROM customer WHERE membership_type = "Premium";

-> Output:

customer_id	name	email	phone	membership_type
2	Prakash	prakashm@gmail.com	9922449438	Premium

Q9) Create a table RENTALS with the given schema.

```
-> CREATE TABLE rentals(  
    rental_id INT PRIMARY KEY,  
    customer_id INT,  
    movie_id INT,  
    rental_date DATE,  
    return_date DATE  
);
```

-> Output:

Field	Type	Null	Key	Default	Extra
rental_id	int	NO	PRI	HULL	
customer_id	int	YES		HULL	
movie_id	int	YES		HULL	
rental_date	date	YES		HULL	
return_date	date	YES		HULL	

Q10) Insert 5 records into the RENTALS table.

```
-> INSERT INTO rentals(rental_id, customer_id, movie_id, rental_date) VALUES(1, 1, 1, "2016-03-01");
```

```
INSERT INTO rentals VALUES(2, 1, 3, "2018-03-01", "2018-04-01");
```

```
INSERT INTO rentals VALUES(3, 1, 5, "2017-08-10", "2017-09-18");
```

```
INSERT INTO rentals VALUES(4, 3, 1, "2014-02-10", "2014-05-30");
```

```
INSERT INTO rentals VALUES(5, 2, 1, "2019-11-07", "2019-12-30");
```

Q11) Add a NOT NULL constraint to the Genre column of the MOVIES table.

-> ALTER TABLE movies

MODIFY COLUMN genre VARCHAR(45) NOT NULL;

-> Output:

Field	Type	Null	Key	Default	Extra
movie_id	int	NO	PRI	NULL	
title	varchar(45)	YES		NULL	
genre	varchar(45)	NO		NULL	
release_date	date	YES		NULL	
rating	int	YES		NULL	
director	varchar(45)	YES		NULL	

Q12) Add a UNIQUE constraint to the Email column in the CUSTOMERS table.

-> ALTER TABLE customer

ADD CONSTRAINT unique_email

UNIQUE customer(email);

-> Output:

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	<small>NULL</small>	
name	varchar(45)	YES		<small>NULL</small>	
email	varchar(45)	YES	UNI	<small>NULL</small>	
phone	varchar(10)	YES		<small>NULL</small>	
membership_type	varchar(45)	YES		<small>NULL</small>	

Q13) Add a foreign key constraint on Movie_ID in the RENTALS table referencing MOVIES(Movie_ID).

-> ALTER TABLE rentals

ADD CONSTRAINT movie_id

FOREIGN KEY (movie_id)

REFERENCES movies(movie_id);

-> Output:

Field	Type	Null	Key	Default	Extra
rental_id	int	NO	PRI	NULL	
customer_id	int	YES		NULL	
movie_id	int	YES	MUL	NULL	
rental_date	date	YES		NULL	
return_date	date	YES		NULL	

Q14) Create an index on the Rating column in the MOVIES table to optimize queries.

-> CREATE INDEX rating_index

ON movies(rating);

-> Output:

Index Name	Type	Index Columns			
PRIMARY	PRIMARY				
rating_index	INDEX	Column	#	Order	Length
		<input type="checkbox"/> movie_id		ASC	
		<input type="checkbox"/> title		ASC	
		<input type="checkbox"/> genre		ASC	
		<input type="checkbox"/> release_date		ASC	
		<input checked="" type="checkbox"/> rating	1	ASC	
		<input type="checkbox"/> director		ASC	

Q15) Find the average rating of all movies in the MOVIES table.

-> SELECT ROUND(AVG(rating),2) AS avg_rating FROM movies;

-> Output:

avg_rating
8.20

Q16) Update all movies with a rating below 5 to change their genre to 'Uncategorized'.

-> UPDATE movies SET genre = "Uncategorized" WHERE rating < 5;

-> Output:

movie_id	title	genre	release_date	rating	director
5	Boss	Uncategorized	2016-12-01	4	Rohit Shetty

Q17) Delete all customer records where the Membership_Type is NULL.

-> DELETE FROM customer WHERE membership_type IS NULL;

-> Output:

customer_id	name	email	phone	membership_type
NULL	NULL	NULL	NULL	NULL

Q18) Retrieve customers who rented movies but have not returned them (use WHERE and NULL check).

-> SELECT c.name

FROM customer c

JOIN rentals r

ON c.customer_id = r.customer_id

WHERE r.return_date IS NULL;

-> Output:

name
Pratyush

Q19) Find the most rented movie(s) and the number of times they were rent.

-> SELECT m.title AS movie, COUNT(r.movie_id) AS no_of_times_rented

FROM movies m

JOIN rentals r

ON m.movie_id = r.movie_id

GROUP BY r.movie_id;

-> Output:

movie	no_of_times_rented
Action Jackson	3
One Piece Film Red	1
Boss	1

C) Bank

Q) Schema Creation.

-> CREATE DATABASE bank;

USE bank;

```
CREATE TABLE customers(  
    customer_id INT PRIMARY KEY,  
    name VARCHAR(50),  
    account_no VARCHAR(20),  
    phone VARCHAR(15),  
    balance DECIMAL(12, 2)  
);
```

```
-> CREATE TABLE transactions(  
    transaction_id INT PRIMARY KEY,  
    account_no VARCHAR(20),  
    transaction_date DATE,  
    amount DECIMAL(12, 2),  
    type VARCHAR(10)  
);
```

-> Output:

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	NULL	
name	varchar(50)	YES		NULL	
account_no	varchar(20)	YES		NULL	
phone	varchar(15)	YES		NULL	
balance	decimal(12,2)	YES		NULL	

Field	Type	Null	Key	Default	Extra
transaction_id	int	NO	PRI	NULL	
account_no	varchar(20)	YES		NULL	
transaction_date	date	YES		NULL	
amount	decimal(12,2)	YES		NULL	
type	varchar(10)	YES		NULL	

Q) Insert rows.

```
-> INSERT INTO customers VALUES(1, "Jane Doe", "123", "8769012351", 60000);  
INSERT INTO customers VALUES(2, "Sahil", "124", "9922449438", 4000);  
INSERT INTO customers VALUES(3, "Pratyush", "125", "7666234983", 70000);  
INSERT INTO customers VALUES(4, "Aarya", "126", "9545447745", 40000);  
INSERT INTO customers VALUES(5, "Kunal", "127", "8678121201", 100000);
```

```
-> INSERT INTO transactions VALUES(1, "123", "2024-03-01", 30000, "Deposit");  
INSERT INTO transactions VALUES(2, "124", "2022-12-10", 2000, "Withdraw");  
INSERT INTO transactions VALUES(3, "125", "2023-12-01", 120000, "Deposit");  
INSERT INTO transactions VALUES(4, "126", "2020-08-17", 1000, "Withdraw");  
INSERT INTO transactions VALUES(5, "123", "2024-05-01", 50000, "Deposit");  
INSERT INTO transactions VALUES(6, "125", "2023-08-17", 10000, "Withdraw");  
INSERT INTO transactions VALUES(7, "127", "2023-09-18", 30000, "Withdraw");
```

Q1) Display all details of customers with a balance greater than 50,000.

-> SELECT * FROM customers WHERE balance > 50000;

-> Output:

customer_id	name	account_no	phone	balance
1	Jane Doe	123	8769012351	60000.00
3	Pratyush	125	7666234983	70000.00
5	Kunal	127	8678121201	100000.00

Q2) Retrieve the names and phone numbers of all customers.

-> SELECT name, phone FROM customers;

-> Output:

name	phone
Jane Doe	8769012351
Sahil	9922449438
Pratyush	7666234983
Aarya	9545447745
Kunal	8678121201

Q3) Display the transaction details for a specific customer named "Jane Doe".

-> SELECT * FROM customers WHERE name = "Jane Doe";

-> Output:

customer_id	name	account_no	phone	balance
1	Jane Doe	123	8769012351	60000.00

Q4) List all transactions made on or after December 1, 2023.

-> SELECT * FROM transactions WHERE transaction_date >= "2023-12-01";

-> Output:

transaction_id	account_no	transaction_date	amount	type
1	123	2024-03-01	30000.00	Deposit
3	125	2023-12-01	120000.00	Deposit
5	123	2024-05-01	50000.00	Deposit

Q5) Find all customers with an account balance less than 5,000.

-> SELECT * FROM customers WHERE balance < 5000;

-> Output:

customer_id	name	account_no	phone	balance
2	Sahil	124	9922449438	4000.00

Q6) Create an index on the Type column in the TRANSACTIONS table.

-> CREATE INDEX type_index

ON transactions(type);

-> Output:

Index Name	Type	Index Columns			
PRIMARY	PRIMARY				
type_index	INDEX				
		Column	#	Order	Length
		<input type="checkbox"/> transaction_id		ASC	
		<input type="checkbox"/> account_no		ASC	
		<input type="checkbox"/> transaction_date		ASC	
		<input type="checkbox"/> amount		ASC	
		<input checked="" type="checkbox"/> type	1	ASC	

Q7) Update the balance of all customers who made deposits greater than 1,00,000.

```
-> UPDATE customers SET balance = balance + (  
    SELECT SUM(t.amount) FROM transactions t  
    WHERE t.account_no = customers.account_no  
    AND t.type = "Deposit"  
    AND t.amount > 100000  
)  
WHERE account_no IN (  
    SELECT DISTINCT account_no  
    FROM transactions  
    WHERE type = "Deposit"  
    AND amount > 100000  
);
```

-> Output:

customer_id	name	account_no	phone	balance
1	Jane Doe	123	8769012351	60000.00
2	Sahil	124	9922449438	4000.00
3	Pratyush	125	7666234983	190000.00
4	Aarya	126	9545447745	40000.00
5	Kunal	127	8678121201	100000.00

Q8) Find the total transaction amount for each customer (use GROUP BY).

```
-> SELECT c.name, SUM(t.amount) AS total_transaction_amount  
FROM customers c  
JOIN transactions t  
ON c.account_no = t.account_no  
GROUP BY c.name;
```

-> Output:

name	total_transaction_amount
Jane Doe	80000.00
Sahil	2000.00
Pratyush	130000.00
Aarya	1000.00
Kunal	30000.00

Q9) Display the first three characters of each customer's name (use a single-row function).

-> `SELECT LEFT(name, 3) AS first_three_characters FROM customers;`

-> Output:

first_three_characters
Jan
Sah
Pra
Aar
Kun

Q10) Create a savepoint after updating the balance of a specific customer.

-> START TRANSACTION;

SELECT * FROM customers WHERE name = "Jane Doe";

SAVEPOINT before_update;

UPDATE customers SET balance = 50000 WHERE name = "Jane Doe";

SAVEPOINT after_update;

SELECT * FROM customers WHERE name = "Jane Doe";

-> Output:

customer_id	name	account_no	phone	balance
1	Jane Doe	123	8769012351	60000.00

customer_id	name	account_no	phone	balance
1	Jane Doe	123	8769012351	50000.00

Q11) Find the average transaction amount for all withdrawals made in December 2023.

```
-> SELECT ROUND(AVG(amount), 2) AS avg_transaction_amt  
FROM transactions WHERE transaction_date  
BETWEEN "2023-01-01" AND "2023-12-30"  
AND type = "Withdraw";
```

-> Output:

avg_transaction_amt
20000.00

Q12) Add a NOT NULL constraint to the Phone column in the CUSTOMERS table.

-> ALTER TABLE customers

MODIFY COLUMN phone VARCHAR(15) NOT NULL;

-> Output:

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	NULL	
name	varchar(50)	YES		NULL	
account_no	varchar(20)	YES		NULL	
phone	varchar(15)	NO		NULL	
balance	decimal(12,2)	YES		NULL	

Q13) Display the details of customers with transactions exceeding 1,00,000 (use JOIN).

```
-> SELECT c.customer_id, c.name, c.account_no, c.phone, c.balance  
FROM customers c  
JOIN transactions t  
ON c.account_no = t.account_no  
WHERE t.amount > 100000;
```

-> Output:

customer_id	name	account_no	phone	balance
3	Pratyush	125	7666234983	190000.00

Q14) Find the customer names along with their total transaction amounts (use GROUP BY).

-> SELECT c.name, SUM(t.amount) AS total_transaction_amounts

FROM customers c

JOIN transactions t

ON c.account_no = t.account_no

GROUP BY c.name;

-> Output:

name	total_transaction_amounts
Jane Doe	80000.00
Sahil	2000.00
Pratyush	130000.00
Aarya	1000.00
Kunal	30000.00

Q15) Delete records of customers with no transactions in the last year.

-> DELETE FROM customers

WHERE account_no NOT IN (

SELECT DISTINCT account_no

FROM transactions

WHERE transaction_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)

);

-> Output:

customer_id	name	account_no	phone	balance
1	Jane Doe	123	8769012351	50000.00