Assignment – 8

Prakash Manikanta Irrinki
Prakashnaidu9494@gmail.com

Assignment 1:

Write a SELECT query to retrieve all columns from a 'customers' table, and modify it to return only the customer's name and email address for customers in a specific city.

select * from customerinfo;

CustomerId	CustomerName	Contact No	city	email
8241	Teja	9876543210	Hyderabad	teja123@gmail.com
8242	Pavan	9792742853	Chennai	pavan54@gmail.com
8243	Prakash	5333353664	Vizag	prakash2002@gmail.com
8244	Harish	9454469393	Bangalore	harish56@gmail.com
8245	Mani	6836489456	Hyderabad	mani1568@gmail.com

select customername,email from customerinfo where city='Hyderabad';

CustomerName	email
Teja	teja123@gmail.com
Mani	mani1568@gmail.com

select customername, email from customers where city='Chennai';

CustomerName	email
Pavan	pavan54@gmail.com

select customername,email from customers where city='Mumbai';

Empty set

Assignment 2:

Craft a query using an INNER JOIN to combine 'orders' and 'customers' tables for customers in a specified region, and a LEFT JOIN to display all customers including those without orders.

select c. customername, c. city, o. orderitem, o. price from customers c inner join orders o on (c. customerid=o. customerid);

customername	city	orderitem	price
Teja	Hyderabad	Mobile	23500
Pavan	Chennai	Laptop	64500
Teja	Hyderabad	Shoe	2000
Prakash	Vizag	Watch	4600

select c. customername, o. orderitem, o. price,o.orderdate from customers c inner join orders o on (c.customerid=o.customerid);

select c.customername,c.contactno,c.city,o.orderitem from customers c inner join orders o on (c.customerid=o.customerid) where OrderItem='shoe';

select c. customername, c. contactno, c. city, o.orderitem from customers c inner join orders o on (c. customerid=o.customerid) where city='vizag';

select c. customerid, c.customername,c.city,o.orderitem,o.price from customers c left join orders o on (c. customerid=o. customerid);

customerid	customername	city	orderitem	price
8241	Teja	Hyderabad	Shoe	2000
8241	Teja	Hyderabad	Mobile	23500
8242	Pavan	Chennai	Laptop	64500
8243	Prakash	Vizag	Watch	4600
8244	Harish	Bengaluru	NULL	NULL
8245	Mani	Hyderabad	NULL	NULL

select c. customername, o. orderitem,o.price,o.orderdate from customers c left join orders o on (c. customerid=o.customerid);

select c. customername, c.contactno,c.city,o.orderitem from customers c left join orders o on (c. customerid=o.customerid) where OrderItem='shoe';

select c.customername,c.contactno,c.city,o.orderitem from customers c left join orders o on (c.customerid=o.customerid) where city='vizag';

Assignment 3:

Utilize a subquery to find customers who have placed orders above the average order value, and write a UNION query to combine two SELECT statements with the same number of columns.

--Sub query a query inside another query

select c. customerid, c. customername, o. orderid, o. OrderItem, orderable from customerinfo c inner join orderinnfo o on c. CustomerId=o.CustomerId where orderable > (select avg (order value) from orders);

customerid	customername	orderid	orderItem	order value
8242	Pavan	128	Laptop	17
8241	Teja	129	Shoe	25

select c. customername, o. orderid, o. OrderItem, o. price from customerinfo c inner join orders o on c. CustomerId=o.CustomerId where o.price > (select min(price) from orders);

customername	orderid	orderItem	price
Teja	121	Mobile	23500
Pavan	128	Laptop	64500
Prakash	130	Watch	4600

--Union will execute two selected queries

select c. customername, c. city, o. orderitem, o. price from customerinfo c inner join orders o on (c. customerid=o. customerid)
union
select c. customername, c. contactno, city, o. orderitem from customers c cross join orders o on (c. customerid=o. customerid) where OrderItem='shoe';

customername	city	orderitem	price
Teja	Hyderabad	Mobile	23500
Pavan	Chennai	Laptop	64500
Teja	Hyderabad	Shoe	2000
Prakash	Vizag	Watch	4600

Assignment 4:

Compose SQL statements to BEGIN a transaction, INSERT a new record into the 'orders' table, COMMIT the transaction, then UPDATE the 'products' table, and ROLLBACK the transaction.

start transaction;

insert into orderinfo values(132,8246,'Table','2024-05-16',3000,4);

commit;

select * from orderinfo;

OrderId	CustomerId	orderItem	orderDate	Price	ordervalue
121	8241	Mobile	2024-01-23	23500	5
128	8242	Laptop	2023-12-25	64500	17
129	8241	Shoe	2024-02-07	2000	25
130	8243	Watch	2024-02-16	4600	11
132	8246	Table	2024-05-16	3000	4

start transaction;

update customer set customerid=8245 where customername='sravya';

select * from customer;

CustomerId	CustomerName	Contact No	city	email
8241	Teja	9876543210	Hyderabad	ravi123@gmail.com
8242	Pavan	9792742853	Chennai	laya54@gmail.com
8243	Prakash	5333353664	Vizag	nani2002@gmail.com
8244	Harish	4544693930	Bengaluru	bhavya56@gmail.com
8245	Mani	6836489456	Hyderabad	sravyasravs@gmail.com

rollback;

--After rollback the uncommitted actions will remain same

select *from customer;

CustomerId	CustomerName	Contact No	city	email
3241	Mani	6836489456	Hyderabad	sravyasravs@gmail.com
8241	Teja	9876543210	Hyderabad	ravi123@gmail.com
8242	Pavan	9792742853	Chennai	laya54@gmail.com
8243	Prakash	5333353664	Vizag	nani2002@gmail.com
8244	Harish	4544693930	Bengaluru	bhavya56@gmail.com

Assignment 5:

Begin a transaction, perform a series of INSERTs into 'orders', setting a SAVEPOINT after each, rollback to the second SAVEPOINT, and COMMIT the overall transaction.

set auto commit=0 start transaction;

insert into orders values(134,8246,'sofa','2024-03-15',300000,7);

save point sp1;

insert into orders values (135,8248,'Dressing table','2024-05-23',3000,8);

insert into orders values(140,8249,'TV','2024-05-23',45000,9);
save point sp2;

insert into orders values(138,8243,'fridge','2023-05-16',35000,15);
save point sp3;

select * from orders;

OrderId	CustomerId	OrderItem	OrderDate	Price	ordervalue
121	8241	Mobile	2024-01-23	23500	5
128	8242	Laptop	2023-12-25	64500	17
129	8241	Shoe	2024-02-07	2000	25
130	8243	Watch	2024-02-16	4600	11
132	8246	Table	2024-05-16	3000	4
134	8246	sofa	2024-03-15	300000	7
135	8248	Dressing table	2024-05-23	3000	8
138	8243	fridge	2023-05-16	35000	15
139	8247	bag	2023-05-11	3000	2
140	8249	TV	2024-05-23	45000	9

rollback to save point sp2;

select * from orders;

OrderId	CustomerId	OrderItem	Order Date	Price	ordervalue
121	8241	Mobile	2024-01-23	23500	5
128	8242	Laptop	2023-12-25	64500	17
129	8241	Shoe	2024-02-07	2000	25
130	8243	Watch	2024-02-16	4600	11
132	8246	Table	2024-05-16	3000	4
134	8246	sofa	2024-03-15	300000	7
135	8248	Dressing table	2024-05-23	3000	8
139	8247	bag	2023-05-11	3000	2

commit;

Assignment 6:

Draft a brief report on the use of transaction logs for data recovery and create a hypothetical scenario where a transaction log is instrumental in data recovery after an unexpected shutdown.

Introduction:

Transaction logs are crucial components of database management systems that record all changes made to a database. These logs serve as a reliable source of information for recovering data in the event of system failures or unexpected shutdowns.

Importance of Transaction Logs:

1. Data Integrity: Transaction logs ensure data integrity by recording every transaction before it is committed to the database. This allows for rollbacks or recovery to a specific point in time.

2. Recovery Point: They provide a recovery point in case of system failures, allowing databases to be restored to a consistent state prior to the failure.

3. Performance Monitoring: Transaction logs also aid in performance monitoring and troubleshooting, as they track changes and can identify potential issues.

Hypothetical Scenario:

Imagine a scenario where a large e-commerce company experiences an unexpected server shutdown during a peak shopping period, resulting in potential data loss and customer disruption. However, due to the implementation of transaction logs, the company's database administrator can initiate a successful data recovery process.

Scenario Details:

1. Unexpected Shutdown: The e-commerce platform experiences a sudden server shutdown due to a power outage.

2. Data Loss Concerns: Concerns arise about potential data loss, including ongoing transactions and customer orders that were being processed.

3. Transaction Logs Utilization: The database administrator leverages transaction logs to restore the database to its state just before the shutdown.

4. Recovery Process: By analysing the transaction logs, the administrator identifies the last committed transactions before the shutdown.

5. Database Restoration: Using this information, the administrator restores the database to the point just before the unexpected shutdown, ensuring minimal data loss and maintaining data consistency.

6. Customer Impact Mitigation: The quick recovery minimizes disruption for customers, allowing them to resume their transactions seamlessly.

Conclusion:

Transaction logs play a vital role in data recovery, especially in scenarios of unexpected shutdowns or system failures. By maintaining a record of all database transactions, transaction logs enable organizations to restore data integrity and minimize downtime, ultimately ensuring business continuity and customer satisfaction.