**BUAN 6320.0W2 - Group Project**

**Group No.3**

**Task: Queries with their output**

**Members:           Roopali Reddy Kallem (RXK180054)**

**Nguyen Thanh Nguyen (NXN210036)**

**Nihar Vinnakota (NXV230017)**

**Haritha Jampani (HXJ220031)**

----**Query 1**: Select all columns and all rows from one table (5 points)

SELECT \* FROM customer;

A screenshot of a computer

Description automatically generated

----**Query 2**: Select five columns and all rows from one table (5 points)

SELECT order\_id, customer\_id, total\_quantity, total\_amount, order\_status

FROM orders;

A screenshot of a computer

Description automatically generated

----**Query 3**: Select all columns from all rows from one view (5 points)

SELECT \* FROM OrderDetails;

A screenshot of a computer

Description automatically generated

----**Query 4**: Using a join on 2 tables, select all columns and all rows from the tables without the use of a Cartesian product (5 points)

SELECT \* FROM customer c LEFT OUTER JOIN orders o ON c.customer\_id = o.customer\_id ;

A screenshot of a computer

Description automatically generated

----**Query 5**: Select and order data retrieved from one table (5 points)

SELECT \* FROM orders

ORDER BY delivery\_date ;

A screenshot of a computer

Description automatically generated

----**Query 6**: Using a join on 3 tables, select 5 columns from the 3 tables. Use syntax that would limit the output to 10 rows (5 points)

SELECT c.customer\_id, c.cust\_lname, o.order\_id, o.total\_amount, p.payment\_status

FROM customer c INNER JOIN orders o ON c.customer\_id = o.customer\_id

INNER JOIN payment p ON c.customer\_id = p.customer\_id

limit 10;

A screenshot of a computer

Description automatically generated

----**Query 7**: Select distinct rows using joins on 3 tables (5 points)

SELECT DISTINCT \*

FROM customer c INNER JOIN orders o ON c.customer\_id = o.customer\_id

INNER JOIN payment p ON c.customer\_id = p.customer\_id ;

A screenshot of a computer

Description automatically generated

----**Query 8:** Use GROUP BY and HAVING in a select statement using one or more tables (5 points)

SELECT pr.product\_id, pr.product\_name, s.supplier\_id, s.supplier\_name, pr.product\_price

FROM product pr INNER JOIN supplier s ON pr.product\_id = s.product\_id

GROUP BY pr.product\_id, pr.product\_name, s.supplier\_id, s.supplier\_name, pr.product\_price

HAVING pr.product\_price = '999.00' ;

A screenshot of a computer

Description automatically generated

----**Query 9:** Use IN clause to select data from one or more tables (5 points)

SELECT \* FROM customer

WHERE customer\_id IN ('1001', '1003', '1005');

A screenshot of a computer

Description automatically generated

----**Query 10:** Select length of one column from one table (use LENGTH function) (5 points)

SELECT LENGTH(cust\_address) FROM customer;

A screenshot of a computer

Description automatically generated

----**Query 11**: Delete one record from one table. Use select statements to demonstrate the table contents before and after the DELETE statement. Make sure you use ROLLBACK

----afterwards so that the data will not be physically removed (5 points)

BEGIN;

-- Display data before deletion

SELECT \* FROM Product;

A screenshot of a computer

Description automatically generated

-- Temporarily remove reference from Supplier

DELETE FROM Supplier WHERE Product\_ID = '3003';

DELETE FROM Orders\_Product WHERE Product\_ID = '3003';

-- Delete the product

DELETE FROM Product WHERE Product\_ID = '3003';

-- Display data after deletion

SELECT \* FROM Product;

A screenshot of a computer

Description automatically generated

-- Rollback the transaction

ROLLBACK;

-- Display data after RollBack

SELECT \* FROM Product;

A screenshot of a computer

Description automatically generated

----**Query 12:** Update one record from one table. Use select statements to demonstrate the table contents before and after the UPDATE statement. Make sure you use ROLLBACK

----afterwards so that the data will not be physically removed (5 points)

SELECT \* FROM supplier;

A screenshot of a computer

Description automatically generated

UPDATE supplier

SET supply\_frequency = Quarterly

WHERE supplier\_name = 'Apple Inc.';

SELECT \* FROM supplier;

A screenshot of a computer

Description automatically generated

ROLLBACK;

-- display after rollback

SELECT \* FROM supplier;

A screenshot of a computer

Description automatically generated

--**Advanced Query 1**: Supplier Reliability Report

---This query generates a report on supplier reliability by comparing the estimated and actual delivery dates of products. It involves a sub-query to get the earliest order date per product.

SELECT

s.Supplier\_Name,

p.Product\_Name,

p.Estimated\_Delivery\_Date,

MIN(o.Delivery\_Date) AS Actual\_Delivery\_Date,

CASE

WHEN MIN(o.Delivery\_Date) <= p.Estimated\_Delivery\_Date THEN 'On Time'

ELSE 'Delayed'

END AS Delivery\_Status

FROM Supplier s

JOIN Product p ON s.Product\_ID = p.Product\_ID

JOIN Orders\_Product op ON p.Product\_ID = op.Product\_ID

JOIN Orders o ON op.Order\_ID = o.Order\_ID

GROUP BY s.Supplier\_Name, p.Product\_Name, p.Estimated\_Delivery\_Date

ORDER BY s.Supplier\_Name, Delivery\_Status;

A screenshot of a computer

Description automatically generated

--**Advanced Query 2**: Customer Spending Analysis by Payment Method

SELECT

c.cust\_Fname || ' ' || c.cust\_LName AS Customer\_Name,

p.Payment\_Method,

COUNT(\*) AS Number\_of\_Orders,

SUM(p.Payment\_Amount) AS Total\_Spent,

CASE

WHEN SUM(p.Payment\_Amount) > 500 THEN 'High Spender'

WHEN SUM(p.Payment\_Amount) BETWEEN 200 AND 500 THEN 'Medium Spender'

ELSE 'Low Spender'

END AS Spending\_Category

FROM Payment p

JOIN Customer c ON p.Customer\_ID = c.Customer\_ID

GROUP BY c.cust\_Fname, c.cust\_LName, p.Payment\_Method

ORDER BY Total\_Spent DESC;

A screenshot of a computer

Description automatically generated

--**Advanced Query3** with Subquery: Identifying High Spending Customers

SELECT

c.cust\_Fname || ' ' || c.cust\_LName AS Customer\_Name,

COUNT(p.Payment\_ID) AS Number\_of\_Payments,

SUM(p.Payment\_Amount::numeric) AS Total\_Spending

FROM Customer c

JOIN Payment p ON c.Customer\_ID = p.Customer\_ID

GROUP BY c.Customer\_ID, c.cust\_Fname, c.cust\_LName

HAVING SUM(p.Payment\_Amount::numeric) > (

SELECT AVG(Total\_Amount::numeric)

FROM (

SELECT

Customer\_ID,

SUM(Payment\_Amount::numeric) AS Total\_Amount

FROM Payment

GROUP BY Customer\_ID

) AS SubQuery

)

ORDER BY Total\_Spending DESC;

A screenshot of a computer

Description automatically generated