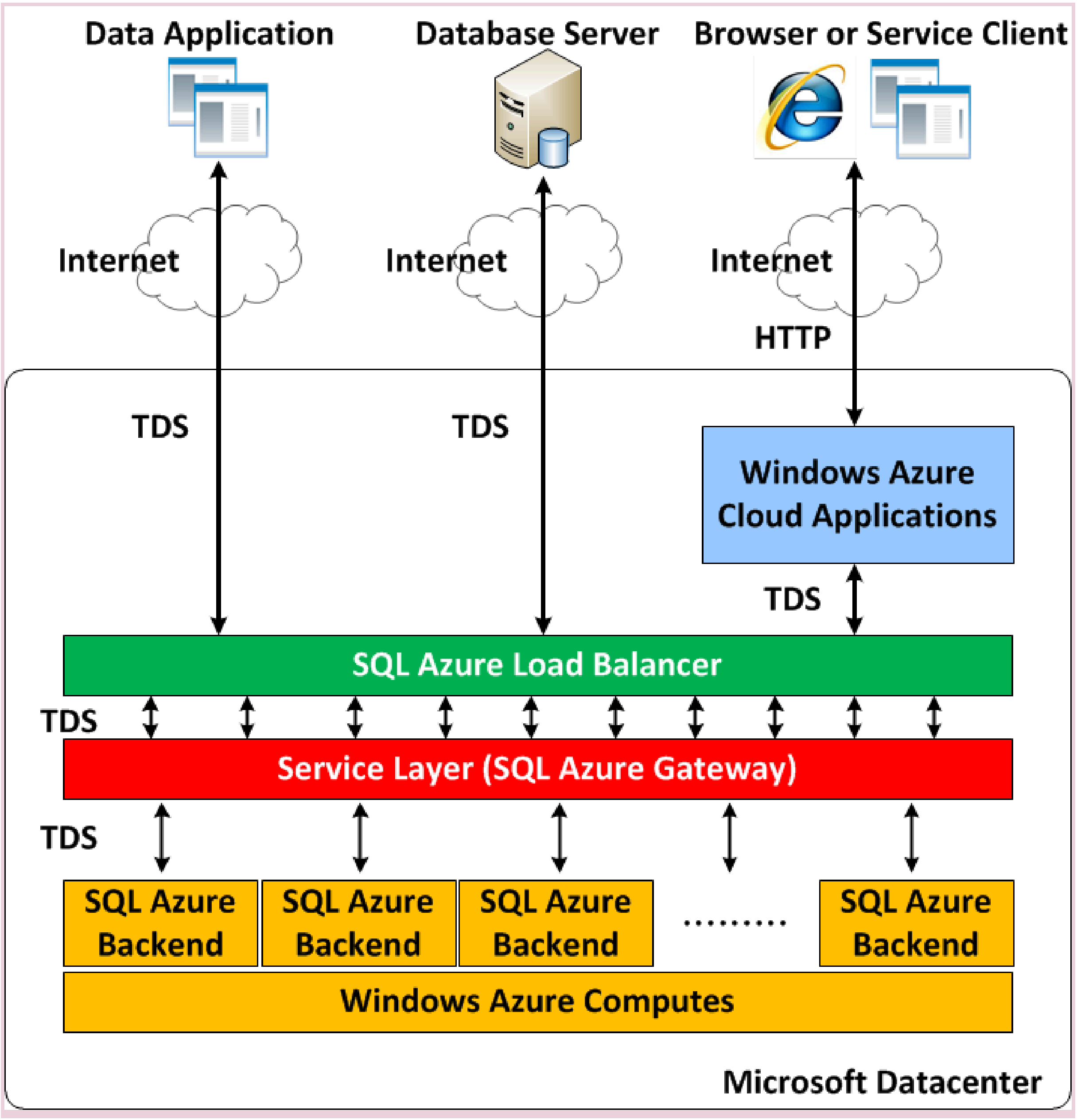
FASTEST REVISION OF SQL:-



-- SQL Complete Course with Examples

-- Section 1: Basic Queries

-- 1.1 SELECT statement

SELECT column1, column2 FROM table_name;

-- 1.2 WHERE clause

```
SELECT * FROM employees WHERE department = 'Sales';

-- 1.3 ORDER BY clause
SELECT name, salary FROM employees ORDER BY salary DESC;

-- Section 2: Data Manipulation
-- 2.1 INSERT statement
INSERT INTO customers (name, email) VALUES ('John Doe',
'john@example.com');

-- 2.2 UPDATE statement
UPDATE products SET price = price * 1.1 WHERE category =
'Electronics';

-- 2.3 DELETE statement
DELETE FROM orders WHERE order_date < '2023-01-01';

-- Section 3: Joins
-- 3.1 INNER JOIN
SELECT orders.order_id, customers.name
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id;

-- 3.2 LEFT JOIN
SELECT employees.name, departments.dept_name
FROM employees
LEFT JOIN departments ON employees.dept_id = departments.dept_id;

-- 3.3 RIGHT JOIN
SELECT products.product_name, categories.category_name
FROM products
RIGHT JOIN categories ON products.category_id =
categories.category_id;

-- Section 4: Aggregate Functions
-- 4.1 COUNT
SELECT department, COUNT(*) as employee_count
FROM employees
```

```
GROUP BY department;

-- 4.2 SUM
SELECT category, SUM(price) as total_value
FROM products
GROUP BY category;

-- 4.3 AVG
SELECT AVG(salary) as average_salary FROM employees;

-- Section 5: Subqueries
-- 5.1 Subquery in WHERE clause
SELECT name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);

-- 5.2 Subquery in FROM clause
SELECT dept_name, avg_salary
FROM (
    SELECT department as dept_name, AVG(salary) as avg_salary
    FROM employees
    GROUP BY department
) AS dept_salaries
WHERE avg_salary > 50000;

-- Section 6: Views
-- 6.1 Creating a view
CREATE VIEW high_value_orders AS
SELECT order_id, customer_id, total_amount
FROM orders
WHERE total_amount > 1000;

-- 6.2 Using a view
SELECT * FROM high_value_orders WHERE customer_id = 101;

-- Section 7: Indexes
-- 7.1 Creating an index
```

```sql
CREATE INDEX idx_last_name ON employees(last_name);

-- Section 8: Transactions
-- 8.1 Basic transaction
BEGIN TRANSACTION;
UPDATE accounts SET balance = balance - 100 WHERE account_id =
1;
UPDATE accounts SET balance = balance + 100 WHERE account_id =
2;
COMMIT;

-- Section 9: Stored Procedures
-- 9.1 Creating a stored procedure
DELIMITER //
CREATE PROCEDURE GetEmployeesByDepartment(IN dept_name
VARCHAR(50))
BEGIN
    SELECT * FROM employees WHERE department = dept_name;
END //
DELIMITER ;

-- 9.2 Calling a stored procedure
CALL GetEmployeesByDepartment('Sales');

-- Section 10: Triggers
-- 10.1 Creating a trigger
CREATE TRIGGER after_order_insert
AFTER INSERT ON orders
FOR EACH ROW
BEGIN
    UPDATE product_inventory
    SET quantity = quantity - NEW.quantity
    WHERE product_id = NEW.product_id;
END;

-- Section 11: Window Functions
-- 11.1 ROW_NUMBER
```

```
SELECT
    employee_name,
    department,
    salary,
    ROW_NUMBER() OVER (PARTITION BY department ORDER BY
salary DESC) as salary_rank
FROM employees;

-- 11.2 RANK
SELECT
    product_name,
    category,
    price,
    RANK() OVER (PARTITION BY category ORDER BY price DESC) as
price_rank
FROM products;

-- Section 12: Common Table Expressions (CTE)
WITH high_salary_employees AS (
    SELECT * FROM employees WHERE salary > 75000
)
SELECT department, COUNT(*) as high_earners
FROM high_salary_employees
GROUP BY department;

-- Section 13: CASE statements
SELECT
    order_id,
    order_total,
    CASE
        WHEN order_total < 100 THEN 'Small Order'
        WHEN order_total BETWEEN 100 AND 1000 THEN 'Medium
Order'
        ELSE 'Large Order'
    END AS order_size
FROM orders;
```

```
-- Section 14: UNION and UNION ALL
SELECT product_name FROM electronics
UNION
SELECT product_name FROM appliances;

-- Section 15: EXISTS
SELECT customer_name
FROM customers c
WHERE EXISTS (
    SELECT 1 FROM orders o
    WHERE o.customer_id = c.customer_id AND o.order_total > 1000
);
```