Software Development Engr. III (Optimum) test

1. Implement a web application using Java, Spring Boot, and Oracle SQL to efficiently read 1 million records from an Oracle table.

-> Here are the key assumptions I made for implementing a Spring Boot + Oracle SQL application to efficiently read 1 million records:

1. Database & Infrastructure

    a. Oracle Database version 12c or higher (using modern JDBC and JPA features).

    b. Use Oracle ROWNUM or ROW_NUMBER() for efficient pagination

    c. Require proper bandwidth supporting bulk data transfer.

    d. Shared Pool can help for large data

2. Application Architecture

    a. Use Spring Boot to use pageSize and pageOffset using Spring pagination service.

    b. Spring Data JPA + native SQL mix can perform optimally (PreparedStatement)

    c. Actuator endpoints monitoring might help in handling large data

    d. ResultSet streaming can help to avoid out of memory errors

    e. Configure JDBC connection with useCursorFetch=true and setFetchSize() with statement and resultset.

    f. Lazy loading can be used for bean creation if it is resource consuming

    g. query or connection timeout can be used for error handling

2. Develop a WebSocket server and client to facilitate real-time bidirectional communication.

-> This depends on what technology we want to use. There is few protocol such as STOMP, raw web socket, JSON, AMQP.

1. STOMP: supports Spring Boot. It supports publish, subscribe and queuing as well.

2. Raw web socket: also support Spring Boot. This is not for wide range use. This is pretty much straight forward messaging system. For high frequency data transfer, it can be very useful.

3. AMQP: Spring Boot supports this thru RabbitMQ. It supports routing technology.

4. JSON: it doesn't support Spring Boot but can be implement manually with spring project. This is used for simple task.

Implementation: Let's assume STOMP protocol.

a. In-memory message broker sufficient for initial scale.
b. Default spring security configuration is enough for this
c. JSON payloads for all messages
d. 10s default heartbeat interval
e. No offline message buffering
f. Implementation simplicity

Note: I have implemented AMQP(RabbitMQ) in my previous experience because of its complex routing system.

3. Write a PL/SQL procedure to summarize the total hours worked by each employee.

-> Let's assume the two table for employee and work hours

TBL_EMPLOYEES

EMP_ID(pk)	EMP_NAME
1	Daniel

TBL_WORK_HOURS

EMP_ID(fk)	WORK_DATE	WORK_HOUR
1	4/17/2025	8

TBL_EMP_HOURS_SUMMARY

EMP_ID	WEEK_START_DATE	TOTAL_HOURS

```
CREATE PROCEDURE summary_employee_workhour AS
V_week_start date := trunc(sysdate, 'IW') - 7;
v_week_end date := trunc(sysdate, 'IW') -1;
BEGIN
```

```
DELETE FROM TBL_EMP_HOURS_SUMMARY WHERE week_start_date = v_week_start;
INSERT INTO TBL_EMP_HOURS_SUMMARY
SELECT e.emp_id, v_week_start AS week_start_date, nvl(SUM(t.hours_worked), 0) AS
total_hours
FROM
tbl_employees e LEFT JOIN tbl_work_hours t ON e.emp_id = t.emp_id AND t.work_date
BETWEEN v_week_start AND v_week_end GROUP BY e.emp_id;
COMMIT;
INSERT INTO job_audit VALUES('HOURS_SUMMARY', SYSDATE, 'Success');
EXCEPTION WHEN OTHERS THEN ROLLBACK;
INSERT INTO job_audit VALUES('HOURS_SUMMARY', SYSDATE, 'Failed: '||SQLERRM);
RAISE;
END;
```

Configure a Scheduler job to automate this report weekly.

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB(
job_name => 'WEEKLY_EMPLOYEE_HOURS_REPORT',
job_type => 'STORED_PROCEDURE',
job_action => 'summary_employee_workhour',
start_date => NEXT_DAY(TRUNC(SYSDATE), 'MONDAY') + 2/24,
repeat_interval => 'FREQ=WEEKLY; INTERVAL=1; BYDAY=MONDAY',
enabled => TRUE
);
END;
```

4. Design an Oracle database schema for an e-commerce website, ensuring data integrity, scalability, and efficient query performance.

-> First, we need to determine the following

Assumption	Justification
Assume B2C business	Most common e-commerce type app
Physical product or digital product	Different requirement
Is operation global or local	Address and currency matters
24/7 support	Higher uptimes require

Let's consider a B2C business

Followings are the example of tables require:

TBL_CUSTOMERS

customer_id	email	password	first_name	last_name	created_at	last_login

```
CREATE TABLE TBL_CUSTOMERS (
customer_id NUMBER GENERATED ALWAYS AS IDENTITTY PRIMARY KEY,
email VARCHAR2(255) NOT NULL UNIQUE,
password VARCHAR2(100),
first_name VARCHAR2(100),
last_name VARCHAR2(100),
created_at TIMESTAMP DEFAULT SYSTIMESTAMP,
last_login TIMESTAM,
CONSTRAINT EMAIL_FORMAT CHECK (REGEXP_LIKE(email, '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$'))) TABLESPACE users_ts;
```

TBL_PRODUCTS

Product_id	sku	name	description	price	cost	Stock_quantity	Category_id	Is_active

```
CREATE TABLE TBL_PRODUCTS (
product_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
sku VARCHAR2(50) NOT NULL UNIQUE,
name VARCHAR2(255) NOT NULL,
description CLOB,
price NUMBER(10,2) NOT NULL,
cost NUMBER(10,2),
stock_quantity NUMBER NOT NULL,
category_id NUMBER,
is_active NUMBER(1) DEFAULT 1,
CONSTRAINT positive_price CHECK (price > 0),
FOREIGN KEY (category_id) REFERENCES product_categories(category_id) )
TABLESPACE users_ts;
```

TBL_ORDERS

order_id	customer_id	order_date	status	total_amount	shipping_address_id	payment_method_id

```
CREATE TABLE TBL_ORDERS (
order_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
customer_id NUMBER NOT NULL,
order_date TIMESTAMP DEFAULT SYSTIMESTAMP,
status VARCHAR2(20) NOT NULL,
total_amount NUMBER(12,2) NOT NULL,
shipping_address_id NUMBER NOT NULL,
payment_method_id NUMBER,
CONSTRAINT fk_customer FOREIGN KEY (customer_id) REFERENCES
customers(customer_id),
CONSTRAINT valid_status CHECK (status IN
('PENDING','PAID','SHIPPED','DELIVERED','CANCELLED')) ) PARTITION BY RANGE
(order_date) ( PARTITION orders_2023 VALUES LESS THAN (TO_DATE('2024-01-01','YYYY-
MM-DD')),
PARTITION orders_2024 VALUES LESS THAN (TO_DATE('2025-01-01','YYYY-MM-DD')) )
TABLESPACE orders_ts;
```

4. **Develop a Java-based multithreaded application to process six tasks (A, B, C, D, E, and F) with the following execution flow:**

 Tasks A & B should run in parallel.

 Tasks C & D should run in parallel after A & B complete.

 The output of A & B should be fed into C & D.

 The output of C & D should be fed into F for final processing.

o Design the application to handle success and failure modes for each task to

ensure robustness and reliability.

-> A java project has been attached to the email. These are simple example withing short period of time. Real time project is much more complicated than these assumptions or codes.