Problem 7:

السيناريو: عندنا جدول اسمه `AppLogs` بيسجل ملايين الـ Requests يومياً. الموبايل أبلكيشن بيعرض داشبورد فيها فلترة سريعة جداً.

Column Name	Data Type	Key / Constraint
log_id	INT	**Primary Key (PK)**
service_name	VARCHAR(50)	Indexed for filtering
status_code	INT	Response status
created_at	DATETIME	Log timestamp
payload	TEXT	Unstructured data

The Query:

```sql
SELECT service_name, created_at, status_code
FROM AppLogs
WHERE service_name = 'AuthService'
  AND created_at >= '2026-01-01'
ORDER BY created_at DESC;
```

المطلوب: اقترح أفضل Composite Index يخلي الكويري دي سريعة ومتجبش معاك time limit لما ت submit

```
CREATE NONCLUSTERED INDEX index_service
ON AppLogs (service_name,created_at,status_code);     The best
```

Fast but more storage space (covering index)

OR

```
CREATE NONCLUSTERED INDEX index_service1
ON AppLogs (service_name,created_at);
```

Lowest speed but less storage space (key lookup)

Problem 8:

السيناريو: الشركة لاحظت إن في Query معينة بطيئة جداً رغم إن في Index موجود فعلاً على العمود المستخدم في الـ `WHERE`.

Column Name	Data Type	Key / Constraint
order_id	INT	**Primary Key (PK)**
customer_id	INT	Foreign Key (Reference)
total_amount	DECIMAL	Precision for financial totals
order_date	DATETIME	Transaction timestamp

(ملاحظة: في Index موجود بالفعل على عمود `order_date` فقط).

المطلوب: ليه الـ Index الحالي مش كفاية وبيجيب time limit؟ وإزاي نعدله بحيث الـ Database متضطرش تروح للجدول الأصلي (Data Pages) خالص؟

Because the index doesn't include all columns used in the select

This leads to key lookup to fetch missing columns from the data pages

To solve this create a covering composite index includes all columns used in select, where, join, order by ...

So the database can retrieve all data without accessing the data pages

Problem 9:

السيناريو: البيزنس عاوز يعمل سيستم عروض سريعة (Flash Sales). المطلوب عمل **Stored Procedure** بتعدل أسعار المنتجات في تصنيف معين (Category) بنسبة خصم محددة، بس مع "صمام أمان" عشان الشركة متخسرش.

Column Name	Data Type	Key / Constraint
product_id	INT	**Primary Key (PK)**
product_name	VARCHAR(100)	Product Name
category_id	INT	Foreign Key (Reference)
price	DECIMAL(10,2)	Current selling price
min_price	DECIMAL(10,2)	Minimum allowed price

The Task:

كتابة Procedure اسمها `sp_ApplyCategoryDiscount` تاخد `CatID@` و `DiscountPercent@`.

المطلوب: الـ Procedure لازم تنزل السعر بالنسبة المطلوبة، بس بشرط إن السعر الجديد ما **يقلش** عن الـ `min_price`. لو قل، السعر يثبت عند الـ `min_price`.

```
Create Procedure sp_ApplyCategoryDiscount
@CatID INT , @DiscountPercent DECIMAL(10,2)
as
update products
set price = CASE
    when (price -(price *(@DiscountPercent/100))) < min_price
    then min_price
    else (price -(price *(@DiscountPercent/100)))
end
where category_id = @CatID;
```

Problem 9:

السيناريو: قسم التسويق عاوز يشوف بيانات العملاء "الـ ثقال" اللي صرفوا مبالغ كبيرة عشان يبعتولهم هدايا، بس إنت كـ Backend Developer لازم تحمي البيانات الحساسة ومينفعش يشوفوا الـ Passwords.

Column Name	Data Type	Key / Constraint
customer_id	INT	**Primary Key (PK)**
name	VARCHAR(100)	Customer Full Name
email	VARCHAR(100)	Contact Info
password	VARCHAR(255)	Sensitive Data (Hidden)
total_spent	DECIMAL(10,2)	Total of all orders

(ملاحظة: الـ `total_spent` بيتحسب من جدول الـ Orders اللي فيه الـ `total_amount`).

The Task:

عمل **View** اسمها `v_VIPCustomers` بتعرض (اسم العميل، إيميله، وإجمالي اللي صرفه).

المطلوب: الـ View تظهر فقط العملاء اللي إجمالي مشترياتهم **أكبر من 5000**، وبترتيب من الأكثر إنفاقاً للأقل.

```sql
Create view VIPCustomers_v as
(
    select c.name,c.email,o.total_spent
    from Customers c join Orders o
    on c.customer_id = o.customer_id
    where o.total_spent>5000
);

select * from VIPCustomers_v
order by total_spent desc;
```