

## Task 3

This task is extremely important to solve it yourself with no or minimal AI assistance cause excessively using AI would cost you minus -5 points...

### Problem 1

#### Not Boring Movies

#Note :

عشان توصل للأرقام الفردية، فكر في "باقي القسمة" او ال modular '%' هو ال Operator اللي بيعرفنا لو قسمنا الرقم على 2 هيتبقى كام؟ لو قدرت تحدد الباقي ده، هتقدر بسهولة تفرق بين ال ID الزوجي والفردى.

### Problem 2

#### Project Employees |

#Note :

#### Official syntax of the round function

```
ROUND ( numeric_expression , length [ , function ] )  
-- Returns a numeric value, rounded to the specified length or precision.
```

SQL

- معامل الطول (length): \* لو موجب: بيقرّب لعدد الخانات العشرية (مثلاً 2 يعني رقمين بعد العلامة).
  - لو سالب: بيقرّب للأرقام اللي "على شمال" العلامة العشرية (مثلاً -1 بيقرّب لأقرب 10).
- المعامل التالت (function): ده اختياري.
  - لو 0 (أو سبته فاضي): بيعمل تقريب (Rounding) حقيقي.
  - لو أي رقم تاني (زي 1): بيعمل بتر (Truncation)، يعني بيقص الأرقام اللي بعد العلامة من غير تقريب.
- النوع (Data Type): النتيجة دايمًا بتطلع بنفس ال Data Type بتاع الرقم اللي دخلته في الأول.

### Problem 3

#### Monthly Transactions |

#Note :

- فكر ازاى تتعامل مع ال Formatting بحيث إن المطلوب عرض الشهر بصيغة YYYY-MM والجدول فيه تاريخ كامل.
- في SQL Server، ابحثوا عن دالة FORMAT () وإزاى تستخدموها عشان تطلعوا "الشهر والسنة" فقط من عمود ال trans\_date. دي هتكون أول عمود في ال SELECT وأساس ال GROUP BY.

## (Conditional Aggregation)

دي أهم تريك في المسألة! إزاي تحسب عدد المبالغ "المقبولة" (approved) بس، في نفس الوقت اللي بتحسب فيه الإجمالي كله؟

- فكروا في استخدام جملة **CASE WHEN** جوه دالة الـ **SUM** أو الـ **COUNT**.
- مثلاً: "لو الحالة approved احسب القيمة، ولو لأ احسبها صفر". الحركة دي بتخليك تعمل فلترة "داخلية" لكل عمود لوحده من غير ما تأثر على باقي النتائج في الـ **WHERE**.

## Problem 4

### Number of unique subjects taught by each teacher

"ركزوا في الـ Schema بتاع الجدول.. هتلاقوا إن الـ **dept\_id** موجود بس عشان يشتكم! المهم هو العلاقة بين المدرس والمادة.. ورونا هتعملوها بـ Query من كام سطر!"

## Problem 5

### Employees whose manager left the company

المطلوب هنا هو الوصول لمدير "موجود كـ ID" في خانة الـ **manager\_id** لكن "مالوش وجود" كـ **employee\_id** في نفس الجدول (لأنه اتمسح).

المسألة دي تطبيق عملي ممتاز على الـ **Filtering with Subqueries**

## Problem 6

### Exchange Seats

- فكر في الأرقام الفردية والزوجية. الملاحظ هنا إن الطالب اللي الـ **id** بتاعه فردي بياخد الرقم الزوجي اللي بعده ( $id+1$ )، والطالب اللي الـ **id** بتاعه زوجي بياخد الرقم الفردي اللي قبله ( $id-1$ ).
- التكنيك: استخدم جملة **CASE WHEN** عشان تعيد تعريف عمود الـ **id** بناءً على كونه فردي أو زوجي (استخدم الـ **Modulo %** اللي عرفناه قبل كده).

## Problem 7

### 1789. Primary Department for Each Employee

- لو الإجابة "لأ"، يبقى الأفضل تستخدموا **UNION ALL** لأنها أسرع في التنفيذ (Performance-wise) لأن SQL مش هيتعب نفسه يدور على تكرارات يمسحها. أما لو شاكين إن في تكرار وعاوزين نتايج فريدة، يبقى الـ **UNION** هي بطة المشهد.

عشان توصل للموظف اللي ملوش غير قسم واحد:

- فكروا في استخدام **GROUP BY** مع **HAVING** عشان تطلعوا الـ **employee\_id** اللي اكرر مرة واحدة بس في الجدول.

## Part of Index problems (Problem 7)

### Problem 1: The Observability Bottleneck (Composite Indexes)

السيناريو: عندنا جدول اسمه **AppLogs** يسجل ملايين الـ Requests يومياً. الموبايل أبل كيشن  
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:

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

SQL

المطلوب: اقترح أفضل Composite Index يخلي الكويبري دي سريعة ومتجش معاك time limit  
لما ت submit

## Problem 8

### Problem 2: The "Key Lookup" Mystery (Covering Indexes)

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؟

## Part of Logic & Abstraction (Problem 9)

## Problem 9: The "Dynamic Flash Sale" (Stored Procedures)

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

## Part of Security & Reporting (Problem 10)

### Problem 10: The "Marketing VIP Dashboard" (Views)

السيناريو: قسم التسويق عاوز يشوف بيانات العملاء "ال ثقال" اللي صرفوا مبالغ كبيرة عشان يبعثولهم هدايا، بس إنت ك Backend Developer لازم تحمي البيانات الحساسة ومينفعش يشوفوا ال 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: ال `total_spent` بيتحسب من جدول ال `Orders` اللي فيه ال `total_amount`).

### The Task:

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

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