17th Oct

Day 12

SQL subqueries

Q Find employees whos salary is more than avg

```
Select * from emp
where salary >
(select avg(saley) from emp)
```

Q Find customess who never order anything.

```
//Select name from Customers C
where not exists (
    Select (1, neme, 4, *) from orders O
    wher   O.customerid = C.id)
```

OR

```
//Select name from Customers C
left join orders O on C.id=O.customerid
where O.id IS NULL
  or O.customerid is NULL.
```

18th Oct
Day 13

SQL

1873 Calculate Special Bonus

FASTER →

```
Select empid,
 Case
 when empid%2!=0 and name not like 'M%'
 then salary
 else 0
End as Bonus
from Employees
ordre by empid asc.
```

- Reads table once   OR

```
Select empid, salary from employees
where empid%2!=0 and name not like 'M%'

union

Select empid, 0 from employees
where empid%2=0 OR name like 'M%'
order by empid asc
```

- Reads table Twice

21 Oct
Day 16

Customers who bought Product 'A', 'B' not 'C'

- Every column which is in select AND not in ~~Group~~ Aggregate func
  ↳ Has to be added in select

- Select c.customerid, c.~~product~~customername
  from Customers c join Orders O on c.customerid = O.customerid
  group by c.customerid, c.customername
  ~~having CASE when sum('product name') = 'A' Else 0~~ ~~Then 1 ELSE 0~~
  ~~when sum('product_name') = 'B' Then 1 Else 0~~
  having
  SUM(Case when O.productname = 'A' Then 1 Else 0 End) > 0
  AND sum (Case when O.product_name = 'B' Then 1 Else 0 End) > 0
  AND sum (case when O.product_name = 'C' Then 1 Else 0 End) = 0
  order by c.customerid.

Having ~~Case~~ SUM(Case when O.productname = 'A' Then 1 Else 0) >0
AND
>0
=0

Day 16 SQL

1112 highest grade for each student

```
select student_id, min(course_id), grade
where (student_id, grade)   IN
~~join~~
(select student id, max(grade) from Enrollments
groupby student_id) ~~[illegible]~~

group by student id, grade
orderby student id
```

OR

```
       studentid, courseid, grade
Select * from (
    select *, rank() over (Partition by studentid
                           orderby grade desc,
                           courseid asc)
    As rank from enrollments)
where rank=1
orderby studentid
```