1.

```
SELECT id, (CASE
   WHEN p_id is null THEN "Root"
   WHEN id IN (SELECT p_id FROM tree WHERE p_id IS NOT NULL) THEN "Inner"
   ELSE "Leaf"
END) AS Type FROM tree AS t1;
```

2.

```
SELECT merchant,
        SUM(CASE WHEN payment_mode = 'cash' THEN amount ELSE 0 END) AS
cash_amount,
        SUM(CASE WHEN payment_mode = 'online' THEN amount ELSE 0 END) AS
online_amount
FROM payments_data GROUP BY merchant;
```

3.

NULL SAFE JOIN

```
select * from tableA inner join tableB on tableA.colA = tableB.colB or
(tableA.colA is null and tableB.colB is null)
```

NULL SAFE OPERATOR in MYSQL

```
select * from tableA inner join tableB on tableA.colA <=> tableB.colB
```

4.

```
Select student_id,
        sum(case when subject = 'ENGLISH' then marks else 0 end) as ENGLISH,
        sum(case when subject = 'SCIENCE' then marks else 0 end) as SCIENCE,
        sum(case when subject = 'MATHS' then marks else 0 end) as MATHS
From marks_date group by student_id;
```

5.

```
Select * from number_pairs t1 where NOT EXISTS ( select * from number_pairs t2 where t1.B=t2.A
and t1.A=t2.B and t1.A > t2.B);
```

6.

```
Select *,
            row_number() over(partition by sorted_str, price) as rank
        from (Select *,
                        case when city1<city2 then concat(city1,city2)
                        when city1>city2 then concat(city2,city1)
                       end as sorted_str
              from travel_data)
        Where rank = 1;
```

7.

```
Select employee_name,
        department_id,
         employee_salary
from (Select *,
              row_number() over(partition by department_id order by employee_salary desc)  as
        rank
        From employee_salary)
Where rank<=3;
```

8.

Since we consider only the required courses in our GPA calculation, we need to exclude optional courses using *WHERE is_required = TRUE*. We need the average GPA per student per year, so we will GROUP BY both the *student_id* and the *school_year* columns and take the average of the *gpa* column. Lastly, we only keep rows where the student has an average GPA higher than 3.5, which can be implemented using HAVING. Let's put everything together:

```
        SELECT
            student_id,
            school_year,
            AVG(gpa) AS avg_gpa
        FROM gpa_history
        WHERE is_required = TRUE
        GROUP BY student_id, school_year
HAVING AVG(gpa) >= 3.5
```

9.

```
Select t1.student_id,t1.student_name,t2.no_of_classes
From (select student_id,count(1) as no_of_classes from table_2 group by student_id) t2
Inner join table_1 t1
On t1.student_id = t2.student_id
```

11.

```
select count(user_id) from(
select count(distinct date) as c, user_id from table
group by user_id having c>1)
```

12.

```
select distinct t1.user_id,case when t2.user_id is not null then 'TRUE' else 'FALSE' end as
overlap
from table t1
left join table2 t2
on t1.user_id != t2.user_id
and t1.start_date <= t2.end_date
and t1.end_date >= t2.start_date
```