SQL LEVEL C

1)

```
WITH CTE1 AS(
SELECT start_date, ROW_NUMBER() OVER (ORDER BY start_date) AS r
FROM Projects
WHERE start_date NOT IN (SELECT end_date FROM Projects)
),
CTE2 AS(
SELECT end_date, ROW_NUMBER() OVER (ORDER BY end_date) AS r
FROM Projects
WHERE end_date NOT IN (SELECT start_date FROM Projects)
)
SELECT CTE1.start_date,CTE2.end_date
FROM CTE1 JOIN CTE2
ON CTE1.r=CTE2.r
ORDER BY CTE2.end_date-CTE1.start_date ASC, CTE1.start_date;
```

2)

```
SELECT Name FROM
(SELECT S1.ID, F.Friend_ID, S1.Name, S2.Name AS Friend, P1.Salary, P2.Salary AS Friend_salary
 FROM
 Students S1 INNER JOIN Friends F
 ON
 S1.ID=F.ID
INNER JOIN Students S2
 ON
 F.Friend_ID=S2.ID
INNER JOIN Packages P1
 ON
 S1.ID=P1.ID
INNER JOIN Packages P2
```

```
 ON
 S2.ID=P2.ID
ORDER BY S1.ID) AS A
WHERE Salary<Friend_salary
ORDER BY Friend_salary;
```

3)

```
SELECT X1, Y1 FROM
(SELECT F1.X AS X1, F1.Y AS Y1, F2.X AS X2, F2.Y AS Y2 FROM Functions F1
INNER JOIN Functions F2 ON F1.X=F2.Y AND F1.Y=F2.X
ORDER BY F1.X) AS A
GROUP BY X1, Y1
HAVING COUNT(X1)>1 OR X1<Y1
ORDER BY X1;
```

4)

```
SELECT con.contest_id, hacker_id, name,
SUM(total_submissions) AS sum_submissions,
SUM(total_accepted_submissions) AS sum_accepted_submissions,
SUM(total_views) AS sum_views,
SUM(total_unique_views) AS sum_unique_views
FROM contests AS con
INNER JOIN colleges AS col
ON con.contest_id = col.contest_id
INNER JOIN challenges AS cha
ON col.college_id = cha.college_id
LEFT JOIN (
   SELECT
   challenge_id,
   SUM(total_views) AS total_views,
   SUM(total_unique_views) AS total_unique_views
```

```
    FROM view_stats
    GROUP BY challenge_id
) AS vs
ON cha.challenge_id = vs.challenge_id
LEFT JOIN (
    SELECT challenge_id,
    SUM(total_submissions) AS total_submissions,
    SUM(total_accepted_submissions) AS total_accepted_submissions
    FROM submission_stats
    GROUP BY challenge_id
) AS ss
ON cha.challenge_id = ss.challenge_id
GROUP BY con.contest_id, hacker_id, name
HAVING sum_submissions > 0 AND sum_accepted_submissions > 0 AND sum_views > 0 AND
sum_unique_views > 0
ORDER BY con.contest_id
```

5)

```
SELECT s1.submission_date, COUNT(DISTINCT s1.hacker_id),
(SELECT hacker_id FROM submissions s3 WHERE s3.submission_date=s1.submission_date GROUP BY
hacker_id
ORDER BY count(submission_id) DESC, MIN(hacker_id) ASC LIMIT 1) AS h_id,
(SELECT h.name FROM hackers h WHERE h.hacker_id = h_id)
FROM submissions s1
WHERE  DATEDIFF(s1.submission_date, '2016-03-01')+1  = (SELECT COUNT(DISTINCT
s2.submission_date) FROM submissions s2
WHERE s2.submission_date <= s1.submission_date AND s2.hacker_id = s1.hacker_id)
GROUP BY s1.submission_date;
```

6)

```
SELECT ROUND(MAX(LAT_N)-MIN(LAT_N) + MAX(LONG_W)-MIN(LONG_W),4) FROM Station;
```

7)

```
declare @maxn int = 1000;
declare @start int = 4;
declare @max_prime int = (select cast(sqrt(@maxn) as int));
declare @primes table (prime int);
insert @primes(prime) values (2), (3);

while @start <= @maxn
begin
   if (select count(*) from @primes) =
   (select count(*) from (select 1 as n from @primes where @start % prime != 0) a )
      insert @primes(prime) values (@start);
   set @start = @start+1;
end

select string_agg(prime, '&')
from @primes;
```

8)

```
SELECT MAX(CASE WHEN occupation='doctor' THEN NAME ELSE NULL END) AS doctor,
MAX(CASE WHEN occupation='professor' THEN NAME ELSE NULL END) AS professor,
MAX(CASE WHEN occupation='singer' THEN NAME ELSE NULL END)AS singer,
MAX(CASE WHEN occupation='actor' THEN NAME ELSE NULL END)AS actor
FROM ( select name , occupation, ROW_NUMBER() over (partition by occupation order by name) as rowno
from occupations )as subtb group by rowno order by rowno
```

9)

```
SELECT N,
CASE
WHEN P is NULL THEN 'Root'
```

```
WHEN N in (SELECT P FROM BST) THEN 'Inner'
ELSE 'Leaf'
END
FROM BST
ORDER by N;
```

10)

```
SELECT
    C.company_code,
    C.founder,
    COUNT(DISTINCT E.lead_manager_code),
    COUNT(DISTINCT E.senior_manager_code),
    COUNT(DISTINCT E.manager_code),
    COUNT(DISTINCT E.employee_code)
FROM Company C
  INNER JOIN Employee E
      ON C.company_code = E.company_code
GROUP BY C.company_code, C.founder
ORDER BY C.company_code;
```

11)

```
SELECT Name FROM
(SELECT S1.ID, F.Friend_ID, S1.Name, S2.Name AS Friend, P1.Salary, P2.Salary AS Friend_salary
 FROM
 Students S1 INNER JOIN Friends F
 ON
 S1.ID=F.ID
INNER JOIN Students S2
 ON
 F.Friend_ID=S2.ID
INNER JOIN Packages P1
```

```
 ON
 S1.ID=P1.ID
INNER JOIN Packages P2
 ON
 S2.ID=P2.ID
ORDER BY S1.ID) AS A
WHERE Salary<Friend_salary
ORDER BY Friend_salary;
```

12)

```
SELECT job_family,
SUM(CASE WHEN location='India' THEN cost ELSE 0 END) AS india_cost,
SUM(CASE WHEN location='International' THEN cost ELSE 0 END) AS international_cost,
(SUM(CASE WHEN location='India' THEN cost ELSE 0 END)/SUM(CASE WHEN location='International' THEN cost ELSE 0 END))*100
AS cost_ratio_percentage
FROM job_family_costs
GROUP BY Job_family;
```

13)

```
SELECT moth,bu,cost,revenue,(cost/revenue) AS cost_revenue_ratio
FROM bu_costs_revenue;
```

14)

```
SELECT sub_band, headcount, headcount/SUM(headcount)
OVER() *100 AS headcount_percentage
FROM sub_band_headcounts;
```

15)

```
select  * from employee where salary in (select distinct top 5 salary from employee order by salary desc)
```

16)

```
CREATE TABLE temp (
   Id INT,
   name VARCHAR(20),
   lastname VARCHAR(20)
);
INSERT INTO temp VALUES
(1, 'ab', 'cd'),
(2, 'df', 'cx'),
(3, 'sd', 'gh');

UPDATE temp AS t1
SET t1.name = t2.lastname,
   t1.lastname = t2.name
FROM temp AS t1
INNER JOIN temp AS t2 ON t1.Id = t2.Id;

SELECT * FROM temp;

-- DROP TABLE temp;
```

17)

```
-- Create a new login
CREATE LOGIN [YourLoginName] WITH PASSWORD = 'YourPassword';
-- Create a user in the database
USE [YourDatabaseName];
CREATE USER [YourUserName] FOR LOGIN [YourLoginName];
-- Assign the db_owner role to the user
EXEC sp_addrolemember 'db_owner', 'YourUserName';
```

18)

```
SELECT
   YEAR(start_date) AS [year],
   MONTH(start_date) AS [month],
   SUM(salary * DATEDIFF(DAY, start_date, ISNULL(end_date, GETDATE())) /
      (DATEDIFF(DAY, start_date, ISNULL(end_date, GETDATE())) + 1)) AS weighted_avg_cost
FROM
   employee_salary
WHERE
   start_date <= GETDATE() AND (end_date >= GETDATE() OR end_date IS NULL)
GROUP BY
   YEAR(start_date),
   MONTH(start_date)
ORDER BY
   [year],
   [month];
```

19)

```
SELECT CAST(CEILING(AVG(CAST(SALARY AS FLOAT)) - AVG(CAST(REPLACE(SALARY,'0','') AS FLOAT)))
AS INT)
FROM EMPLOYEES;
```

20)

```
INSERT INTO destination_table (column1, column2, ...)
SELECT column1, column2, ... FROM source_table;
```