Indexing Performance Report

This report evaluates the impact of various indexing strategies on the performance of five advanced SQL queries. The primary goal was to determine whether indexes could enhance query execution cost.

Indexing Configurations Tested

The following indexing strategies were applied and tested:

- **Design A**: Create an index on Application(AppliedAt)
- **Design B**: Create a composite index on Application(UserID, AppliedAt)
- **Design C**:

  Create an index on Application(UserID)
  Create an index on Application(JobID)
  Create an index on Job(JobID)

Observed Query Costs

Query	Baseline Cost	Design A Cost	Design B Cost	Design C Cost
1	398	1133	398	398
2	771	794	771	771
3	771	794	794	794
4	402452...	416522...	354519...	416522...
5	445	448	467	448

Analysis and Explanation:

The analysis of the indexing configurations reveals the following insights:

- Design A significantly increased the cost of Query 1, from 398 to 1133, without improving other queries.
- Design B showed no change for Queries 1 and 2 compared to baseline but increased the cost for Query 3 and improved Query 4 by lowering the cost from 402452... to 354519... .
- Design C had behavior similar to Design A in most cases, with no significant improvement; it also matched Design A's high cost for Query 4 (416522...).

Observations:

- Indexing on AppliedAt alone (Design A) resulted in a major performance degradation likely because we found it less efficient compared to baseline.
- The composite index (Design B) was beneficial for the most expensive query (Query 4), suggesting that the combination of UserID and AppliedAt helped in narrowing down results more efficiently for complex subqueries.
- Separate indexes on UserID, JobID (Design C) did not provide significant performance gains over the baseline.

Recommendations:

Based on the results

- Design B (Composite index on UserID and AppliedAt) should be selected as the optimal index design.
- Even though not all queries benefited, the most computationally expensive query (Query 4) showed a large cost reduction.
- We chose Design B because as we can we Design A and C had a cost greater than 400000… for Query 4 but Design had a much lesser cost.
- Design B had a lower cost on average for all the queries compared to Designs A and C.
- Therefore the final index plan chosen for all queries is Design B.

EXPLAIN ANALYZE FOR EACH QUERY:

Query 1:

```
EXPLAIN ANALYZE
SELECT
  a.ApplicationID,
  u.Name,
  j.Title,
  a.AppliedAt
FROM
  Application a
JOIN
  User u ON a.UserID = u.UserID
JOIN
  Job j ON a.JobID = j.JobID
WHERE
  a.ApplicationID IN (
    SELECT a2.ApplicationID
    FROM Application a2
    WHERE a2.AppliedAt >= NOW() - INTERVAL 45 DAY
  )
ORDER BY
  a.AppliedAt DESC;
```

```
mysql>
mysql> EXPLAIN ANALYZE
    -> SELECT
    ->     a.ApplicationID,
    ->     u.Name,
    ->     j.Title,
    ->     a.AppliedAt
    -> FROM
    ->     Application a
    -> JOIN
    ->     User u ON a.UserID = u.UserID
    -> JOIN
    ->     Job j ON a.JobID = j.JobID
    -> WHERE
    ->     a.ApplicationID IN (
    ->         SELECT a2.ApplicationID
    ->         FROM Application a2
    ->         WHERE a2.AppliedAt >= NOW() - INTERVAL 45 DAY
    ->     )
    -> ORDER BY
    ->     a.AppliedAt DESC;
+------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                  |
+------------------------------------------------------------------------------------------+
| -> Sort: a.AppliedAt DESC  (actual time=6.41..6.48 rows=1035 loops=1)
    -> Stream results  (cost=398 rows=345) (actual time=0.252..5.96 rows=1035 loops=1)
        -> Nested loop inner join  (cost=398 rows=345) (actual time=0.241..5.56 rows=1035 loops=1)
            -> Nested loop inner join  (cost=277 rows=345) (actual time=0.22..4 rows=1035 loops=1)
                -> Nested loop inner join  (cost=156 rows=345) (actual time=0.197..2.08 rows=1035 loops=1)
                    -> Filter: (a2.AppliedAt >= <cache>((now() - interval 45 day)))  (cost=35.7 rows=345) (actual time=0.169..0.565 rows=1035 loops=1)
                        -> Table scan on a2  (cost=35.7 rows=1035) (actual time=0.118..0.436 rows=1035 loops=1)
                    -> Single-row index lookup on a using PRIMARY (ApplicationID = a2.ApplicationID)  (cost=0.25 rows=1) (actual time=0.00133..0.00135 rows=1 loops=1035)
                -> Single-row index lookup on u using PRIMARY (UserID = a.UserID)  (cost=0.25 rows=1) (actual time=0.00171..0.00174 rows=1 loops=1035)
            -> Single-row index lookup on j using PRIMARY (JobID = a.JobID)  (cost=0.25 rows=1) (actual time=0.00137..0.00139 rows=1 loops=1035)
 |
+------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```

Query 2:

```
EXPLAIN ANALYZE
SELECT
  u.UserID,
  u.Name,
  j.JobID,
  j.Title,
  a.AppliedAt
FROM
  User u
JOIN
  Application a ON u.UserID = a.UserID
JOIN
  Job j ON a.JobID = j.JobID
WHERE
  a.AppliedAt = (
    SELECT MAX(a2.AppliedAt)
    FROM Application a2
    WHERE a2.UserID = u.UserID
  );
```

```
mysql> EXPLAIN ANALYZE
    -> SELECT
    ->     u.UserID,
    ->     u.Name,
    ->     j.JobID,
    ->     j.Title,
    ->     a.AppliedAt
    -> FROM
    ->     User u
    -> JOIN
    ->     Application a ON u.UserID = a.UserID
    -> JOIN
    ->     Job j ON a.JobID = j.JobID
    -> WHERE
    ->     a.AppliedAt = (
    ->         SELECT MAX(a2.AppliedAt)
    ->         FROM Application a2
    ->         WHERE a2.UserID = u.UserID
    ->     );
+----------------------------------------------------------------------------+
| EXPLAIN                                                                    |
+----------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=771 rows=934) (actual time=0.921..16 rows=1000 loops=1)
    -> Nested loop inner join  (cost=445 rows=934) (actual time=0.841..13 rows=1000 loops=1)
        -> Table scan on u  (cost=118 rows=934) (actual time=0.382..1.61 rows=1008 loops=1)
        -> Filter: (a.AppliedAt = (select #2))  (cost=0.25 rows=1) (actual time=0.0104..0.0111 rows=0.992 loops=1008)
            -> Index lookup on a using UserID (UserID = u.UserID)  (cost=0.25 rows=1) (actual time=0.00373..0.00439 rows=1.03 loops=1008)
            -> Select #2 (subquery in condition; dependent)
                -> Aggregate: max(a2.AppliedAt)  (cost=0.58 rows=1) (actual time=0.00506..0.0051 rows=1 loops=1035)
                    -> Index lookup on a2 using UserID (UserID = u.UserID)  (cost=0.35 rows=1) (actual time=0.00393..0.00454 rows=1.09 loops=1035)
    -> Single-row index lookup on j using PRIMARY (JobID = a.JobID)  (cost=0.25 rows=1) (actual time=0.00273..0.00277 rows=1 loops=1000)
 |
+----------------------------------------------------------------------------+
1 row in set, 1 warning (0.02 sec)
```

Query 3:

EXPLAIN ANALYZE

SELECT

u.UserID,

u.Name,

j.Company,

COUNT(DISTINCT j.JobID) AS NumJobsAtCompany

FROM

User u

JOIN

Application a ON u.UserID = a.UserID

JOIN

Job j ON a.JobID = j.JobID

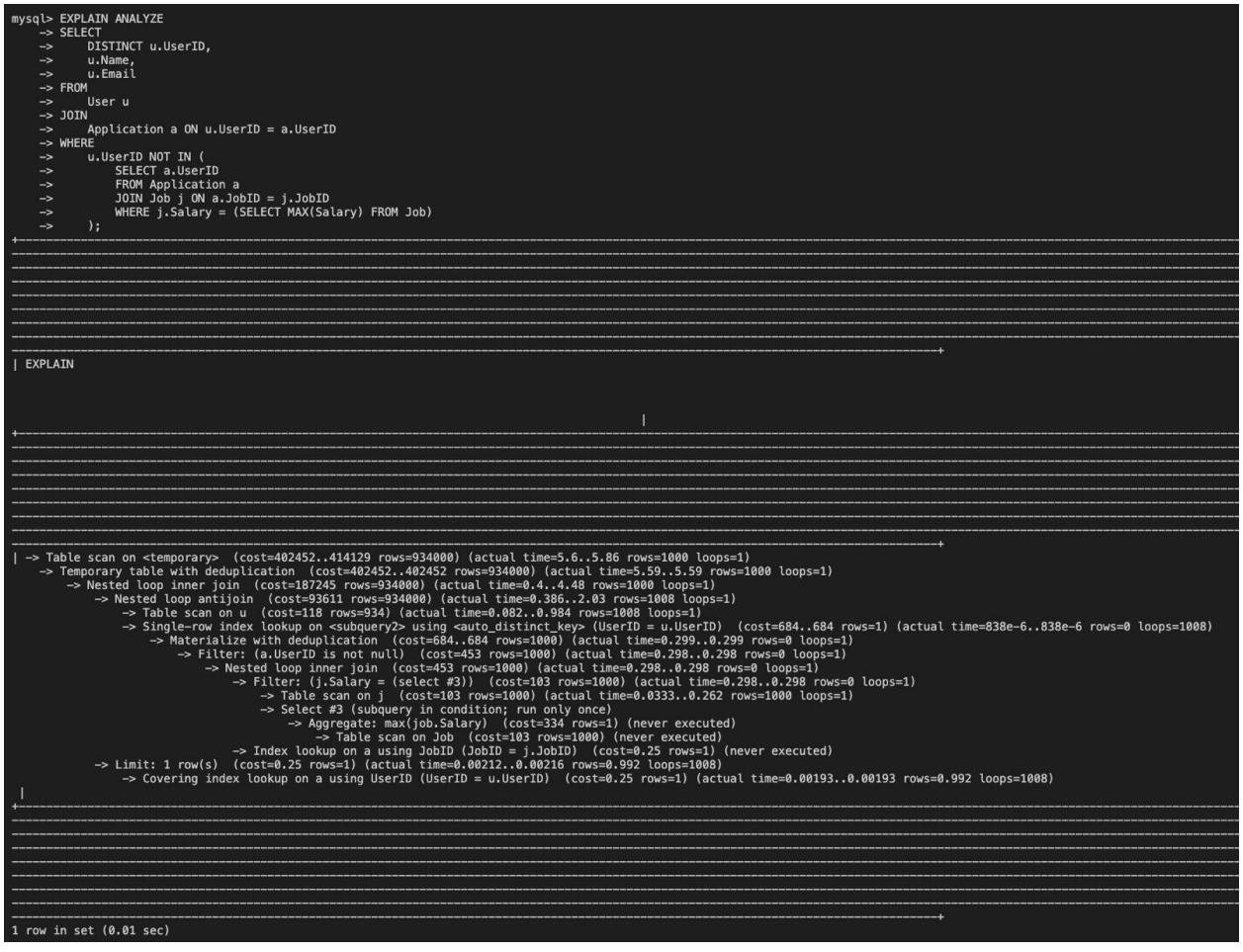
GROUP BY

u.UserID, u.Name, j.Company;

```
mysql> EXPLAIN ANALYZE
    -> SELECT
    ->     u.UserID,
    ->     u.Name,
    ->     j.Company,
    ->     COUNT(DISTINCT j.JobID) AS NumJobsAtCompany
    -> FROM
    ->     User u
    -> JOIN
    ->     Application a ON u.UserID = a.UserID
    -> JOIN
    ->     Job j ON a.JobID = j.JobID
    -> GROUP BY
    ->     u.UserID, u.Name, j.Company;
| EXPLAIN                                                                                  |
| -> Group aggregate: count(distinct job.JobID)  (actual time=9.03..9.61 rows=1033 loops=1)
    -> Sort: u.UserID, u.`Name`, j.Company  (actual time=9.01..9.09 rows=1035 loops=1)
        -> Stream results  (cost=771 rows=934) (actual time=0.202..8.14 rows=1035 loops=1)
            -> Nested loop inner join  (cost=771 rows=934) (actual time=0.196..7.55 rows=1035 loops=1)
                -> Nested loop inner join  (cost=445 rows=934) (actual time=0.185..4.95 rows=1035 loops=1)
                    -> Table scan on u  (cost=118 rows=934) (actual time=0.14..0.802 rows=1008 loops=1)
                    -> Index lookup on a using UserID (UserID = u.UserID)  (cost=0.25 rows=1) (actual time=0.00325..0.00391 rows=1.03 loops=1008)
                -> Single-row index lookup on j using PRIMARY (JobID = a.JobID)  (cost=0.25 rows=1) (actual time=0.00225..0.00229 rows=1 loops=1035)
 |
1 row in set (0.01 sec)
```

Query 4:

```
EXPLAIN ANALYZE
SELECT
  DISTINCT u.UserID,
  u.Name,
  u.Email
FROM
  User u
JOIN
  Application a ON u.UserID = a.UserID
WHERE
  u.UserID NOT IN (
    SELECT a.UserID
    FROM Application a
    JOIN Job j ON a.JobID = j.JobID
    WHERE j.Salary = (SELECT MAX(Salary) FROM Job)
  );
```



Query 5:

```
EXPLAIN ANALYZE
SELECT
  u.UserID,
  COUNT(a.ApplicationID) AS TotalApplications
FROM
  User u
JOIN
  Application a ON u.UserID = a.UserID
GROUP BY
  u.UserID
HAVING
  COUNT(a.ApplicationID) >= (
    SELECT AVG(app_count)
    FROM (
      SELECT COUNT(*) AS app_count
      FROM Application
      GROUP BY UserID
    ) AS sub
  );
```

```
mysql> EXPLAIN ANALYZE
    -> SELECT
    ->     u.UserID,
    ->     COUNT(a.ApplicationID) AS TotalApplications
    -> FROM
    ->     User u
    -> JOIN
    ->     Application a ON u.UserID = a.UserID
    -> GROUP BY
    ->     u.UserID
    -> HAVING
    ->     COUNT(a.ApplicationID) >= (
    ->         SELECT AVG(app_count)
    ->         FROM (
    ->             SELECT COUNT(*) AS app_count
    ->             FROM Application
    ->             GROUP BY UserID
    ->         ) AS sub
    ->     );
+--------------------------------------------------------------------------------+
| EXPLAIN                                                                        |
+--------------------------------------------------------------------------------+
| -> Filter: (count(a.ApplicationID) >= (select #2))  (cost=660 rows=927) (actual time=1.52..4.8 rows=25 loops=1)
    -> Group aggregate: count(a.ApplicationID), count(a.ApplicationID)  (cost=660 rows=927) (actual time=0.572..3.64 rows=1000 loops=1)
        -> Nested loop inner join  (cost=445 rows=934) (actual time=0.555..3.35 rows=1035 loops=1)
            -> Covering index scan on u using PRIMARY  (cost=118 rows=934) (actual time=0.531..0.832 rows=1008 loops=1)
            -> Covering index lookup on a using UserID (UserID = u.UserID)  (cost=0.25 rows=1) (actual time=0.00173..0.00226 rows=1.03 loops=1008)
    -> Select #2 (subquery in condition; run only once)
        -> Aggregate: avg(sub.app_count)  (cost=836..836 rows=1) (actual time=0.923..0.923 rows=1 loops=1)
            -> Table scan on sub  (cost=582..597 rows=1035) (actual time=0.68..0.809 rows=1000 loops=1)
                -> Materialize  (cost=582..582 rows=1035) (actual time=0.678..0.678 rows=1000 loops=1)
                    -> Group aggregate: count(0)  (cost=343 rows=1035) (actual time=0.084..0.494 rows=1000 loops=1)
                        -> Covering index scan on Application using UserID  (cost=105 rows=1035) (actual time=0.0819..0.321 rows=1035 loops=1)
 |
+--------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```

Indexing Plans

Design A: CREATE INDEX idx_appliedat ON Application(AppliedAt);

Query 1:

```
mysql> CREATE INDEX idx_appliedat ON Application(AppliedAt);
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT      a.ApplicationID,     u.Name,     j.Title,     a.AppliedAt FROM      Application a JOIN      User u ON a.UserID = u.UserID JOIN      Job j ON a.JobID = j
obID WHERE      a.ApplicationID IN (         SELECT a2.ApplicationID         FROM Application a2         WHERE a2.AppliedAt >= NOW() - INTERVAL 45 DAY     ) ORDER BY      a.AppliedAt DESC

| EXPLAIN

| -> Sort: a.AppliedAt DESC  (actual time=16.1..16.2 rows=1035 loops=1)
    -> Stream results  (cost=1133 rows=967) (actual time=0.245..15.5 rows=1035 loops=1)
        -> Nested loop inner join  (cost=1133 rows=967) (actual time=0.236..14.5 rows=1035 loops=1)
            -> Nested loop inner join  (cost=794 rows=967) (actual time=0.217..10.9 rows=1035 loops=1)
                -> Nested loop inner join  (cost=456 rows=967) (actual time=0.206..7.35 rows=1035 loops=1)
                    -> Table scan on u  (cost=118 rows=934) (actual time=0.143..1.27 rows=1008 loops=1)
                    -> Index lookup on a using UserID (UserID = u.UserID)  (cost=0.259 rows=1.03) (actual time=0.00481..0.00574 rows=1.03 loops=1008)
                -> Single-row index lookup on j using PRIMARY (JobID = a.JobID)  (cost=0.25 rows=1) (actual time=0.00311..0.00317 rows=1 loops=1035)
            -> Filter: (a2.AppliedAt >= <cache>((now() - interval 45 day)))  (cost=0.25 rows=1) (actual time=0.00303..0.00319 rows=1 loops=1035)
                -> Single-row index lookup on a2 using PRIMARY (ApplicationID = a.ApplicationID)  (cost=0.25 rows=1) (actual time=0.00271..0.00277 rows=1 loops=1035)
 |

1 row in set (0.02 sec)
```

Query 2:

```
mysql> CREATE INDEX idx_appliedat ON Application(AppliedAt);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT      u.UserID,     u.Name,     j.JobID,     j.Title,     a.AppliedAt FROM      User u JOIN      Application a ON u.UserID = a.UserID JOIN      Job j ON a.
JobID = j.JobID WHERE      a.AppliedAt = (         SELECT MAX(a2.AppliedAt)         FROM Application a2         WHERE a2.UserID = u.UserID     );

| EXPLAIN

| -> Nested loop inner join  (cost=794 rows=967) (actual time=0.464..10.8 rows=1000 loops=1)
    -> Nested loop inner join  (cost=456 rows=967) (actual time=0.423..8.73 rows=1000 loops=1)
        -> Table scan on u  (cost=118 rows=934) (actual time=0.296..1.35 rows=1008 loops=1)
        -> Filter: (a.AppliedAt = (select #2))  (cost=0.259 rows=1.03) (actual time=0.00673..0.00719 rows=0.992 loops=1008)
            -> Index lookup on a using UserID (UserID = u.UserID)  (cost=0.259 rows=1.03) (actual time=0.00233..0.00275 rows=1.03 loops=1008)
            -> Select #2 (subquery in condition; dependent)
                -> Aggregate: max(a2.AppliedAt)  (cost=0.601 rows=1) (actual time=0.00295..0.00298 rows=1 loops=1035)
                    -> Index lookup on a2 using UserID (UserID = u.UserID)  (cost=0.362 rows=1.03) (actual time=0.00229..0.00263 rows=1.09 loops=1035)
    -> Single-row index lookup on j using PRIMARY (JobID = a.JobID)  (cost=0.25 rows=1) (actual time=0.00187..0.00189 rows=1 loops=1000)
 |

1 row in set, 1 warning (0.02 sec)
```

Query 3:

```
mysql> DROP INDEX idx_appliedat ON Application;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> CREATE INDEX idx_appliedat ON Application(AppliedAt);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE
    -> SELECT
    ->     u.UserID,
    ->     u.Name,
    ->     j.Company,
    ->     COUNT(DISTINCT j.JobID) AS NumJobsAtCompany
    -> FROM
    ->     User u
    -> JOIN
    ->     Application a ON u.UserID = a.UserID
    -> JOIN
    ->     Job j ON a.JobID = j.JobID
    -> GROUP BY
    ->     u.UserID, u.Name, j.Company;
+------------------------------------------------------------------------------+
| EXPLAIN                                                                      |
+------------------------------------------------------------------------------+
| -> Group aggregate: count(distinct job.JobID)  (actual time=6.05..6.62 rows=1033 loops=1)
    -> Sort: u.UserID, u.`Name`, j.Company  (actual time=6.04..6.11 rows=1035 loops=1)
        -> Stream results  (cost=794 rows=967) (actual time=0.285..5.58 rows=1035 loops=1)
            -> Nested loop inner join  (cost=794 rows=967) (actual time=0.279..5.27 rows=1035 loops=1)
                -> Nested loop inner join  (cost=456 rows=967) (actual time=0.239..3.52 rows=1035 loops=1)
                    -> Table scan on u  (cost=118 rows=934) (actual time=0.163..0.953 rows=1008 loops=1)
                    -> Index lookup on a using UserID (UserID = u.UserID)  (cost=0.259 rows=1.03) (actual time=0.00205..0.00243 rows=1.03 loops=1008)
                -> Single-row index lookup on j using PRIMARY (JobID = a.JobID)  (cost=0.25 rows=1) (actual time=0.00155..0.00157 rows=1 loops=1035)
 |
+------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```

Query 4:

```
mysql> DROP INDEX idx_appliedat ON Application;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> CREATE INDEX idx_appliedat ON Application(AppliedAt);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE
    -> SELECT
    ->     DISTINCT u.UserID,
    ->     u.Name,
    ->     u.Email
    -> FROM
    ->     User u
    -> JOIN
    ->     Application a ON u.UserID = a.UserID
    -> WHERE
    ->     u.UserID NOT IN (
    ->         SELECT a.UserID
    ->         FROM Application a
    ->         JOIN Job j ON a.JobID = j.JobID
    ->         WHERE j.Salary = (SELECT MAX(Salary) FROM Job)
    ->     );
+------------------------------------------------------------------------------+
| EXPLAIN                                                                      |
+------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (cost=416522..428608 rows=966690) (actual time=3.42..3.53 rows=1000 loops=1)
    -> Temporary table with deduplication  (cost=416522..416522 rows=966690) (actual time=3.42..3.42 rows=1000 loops=1)
        -> Nested loop inner join  (cost=193783 rows=966690) (actual time=0.412..2.76 rows=1000 loops=1)
            -> Nested loop antijoin  (cost=96880 rows=966690) (actual time=0.391..1.43 rows=1008 loops=1)
                -> Table scan on u  (cost=118 rows=934) (actual time=0.0825..0.715 rows=1008 loops=1)
                -> Single-row index lookup on <subquery2> using <auto_distinct_key> (UserID = u.UserID)  (cost=704..704 rows=1) (actual time=601e-6..601e-6 rows=0 loops=1008)
                    -> Materialize with deduplication  (cost=704..704 rows=1035) (actual time=0.299..0.299 rows=0 loops=1)
                        -> Filter: (a.UserID is not null)  (cost=465 rows=1035) (actual time=0.297..0.297 rows=0 loops=1)
                            -> Nested loop inner join  (cost=465 rows=1035) (actual time=0.297..0.297 rows=0 loops=1)
                                -> Filter: (j.Salary = (select #3))  (cost=103 rows=1000) (actual time=0.297..0.297 rows=0 loops=1)
                                    -> Table scan on j  (cost=103 rows=1000) (actual time=0.0195..0.26 rows=1000 loops=1)
                                    -> Select #3 (subquery in condition; run only once)
                                        -> Aggregate: max(job.Salary)  (cost=334 rows=1) (never executed)
                                            -> Table scan on Job  (cost=103 rows=1000) (never executed)
                                -> Index lookup on a using JobID (JobID = j.JobID)  (cost=0.259 rows=1.03) (never executed)
            -> Limit: 1 row(s)  (cost=0.25 rows=1) (actual time=0.00115..0.00117 rows=0.992 loops=1008)
                -> Covering index lookup on a using UserID (UserID = u.UserID)  (cost=0.25 rows=1.03) (actual time=0.00104..0.00104 rows=0.992 loops=1008)
 |
+------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

Query 5:

```
mysql> DROP INDEX idx_appliedat ON Application;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> CREATE INDEX idx_appliedat ON Application(AppliedAt);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE
    -> SELECT
    ->     u.UserID,
    ->     COUNT(a.ApplicationID) AS TotalApplications
    -> FROM
    ->     User u
    -> JOIN
    ->     Application a ON u.UserID = a.UserID
    -> GROUP BY
    ->     u.UserID
    -> HAVING
    ->     COUNT(a.ApplicationID) >= (
    ->         SELECT AVG(app_count)
    ->         FROM (
    ->             SELECT COUNT(*) AS app_count
    ->             FROM Application
    ->             GROUP BY UserID
    ->         ) AS sub
    ->     );
+--------------------------------------------------------------------------------+
| EXPLAIN                                                                          |
+--------------------------------------------------------------------------------+
| -> Filter: (count(a.ApplicationID) >= (select #2))  (cost=671 rows=927) (actual time=2.05..5.43 rows=25 loops=1)
    -> Group aggregate: count(a.ApplicationID), count(a.ApplicationID)  (cost=671 rows=927) (actual time=1.11..4.29 rows=1000 loops=1)
        -> Nested loop inner join  (cost=448 rows=967) (actual time=1.1..4.03 rows=1035 loops=1)
            -> Covering index scan on u using PRIMARY  (cost=118 rows=934) (actual time=1.06..1.35 rows=1008 loops=1)
            -> Covering index lookup on a using UserID (UserID = u.UserID)  (cost=0.25 rows=1.03) (actual time=0.00169..0.00233 rows=1.03 loops=1008)
    -> Select #2 (subquery in condition; run only once)
        -> Aggregate: avg(sub.app_count)  (cost=819..819 rows=1) (actual time=0.901..0.901 rows=1 loops=1)
            -> Table scan on sub  (cost=574..589 rows=1000) (actual time=0.654..0.785 rows=1000 loops=1)
                -> Materialize  (cost=574..574 rows=1000) (actual time=0.652..0.652 rows=1000 loops=1)
                    -> Group aggregate: count(0)  (cost=343 rows=1000) (actual time=0.0831..0.455 rows=1000 loops=1)
                        -> Covering index scan on Application using UserID  (cost=105 rows=1035) (actual time=0.0807..0.309 rows=1035 loops=1)
 |
+--------------------------------------------------------------------------------+
1 ro  et (0.01 sec)
```

Design B : CREATE INDEX idx_userid_appliedat ON Application(UserID, AppliedAt);

Query 1:

```
mysql> DROP INDEX idx_appliedat ON Application;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> CREATE INDEX idx_userid_appliedat ON Application(UserID, AppliedAt);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT      a.ApplicationID,     u.Name,     j.Title,     a.AppliedAt FROM      Application a JOIN      User u ON a.UserID = u.UserID JOIN      Job j ON a.JobID = j.JobID WHERE      a.ApplicationID IN (
SELECT a2.ApplicationID         FROM Application a2         WHERE a2.AppliedAt >= NOW() - INTERVAL 45 DAY     ) ORDER BY      a.AppliedAt DESC;

| EXPLAIN

| -> Sort: a.AppliedAt DESC  (actual time=10.4..10.5 rows=1035 loops=1)
    -> Stream results  (cost=398 rows=345) (actual time=0.243..9.95 rows=1035 loops=1)
        -> Nested loop inner join  (cost=398 rows=345) (actual time=0.231..9.26 rows=1035 loops=1)
            -> Nested loop inner join  (cost=277 rows=345) (actual time=0.172..6.58 rows=1035 loops=1)
                -> Nested loop inner join  (cost=156 rows=345) (actual time=0.121..2.69 rows=1035 loops=1)
                    -> Filter: (a2.AppliedAt >= <cache>((now() - interval 45 day)))  (cost=35.7 rows=345) (actual time=0.094..0.595 rows=1035 loops=1)
                        -> Covering index scan on a2 using idx_userid_appliedat  (cost=35.7 rows=1035) (actual time=0.0763..0.445 rows=1035 loops=1)
                    -> Single-row index lookup on a using PRIMARY (ApplicationID = a2.ApplicationID)  (cost=0.25 rows=1) (actual time=0.00184..0.00187 rows=1 loops=1035)
                -> Single-row index lookup on u using PRIMARY (UserID = a.UserID)  (cost=0.25 rows=1) (actual time=0.00357..0.0036 rows=1 loops=1035)
            -> Single-row index lookup on j using PRIMARY (JobID = a.JobID)  (cost=0.25 rows=1) (actual time=0.00218..0.00221 rows=1 loops=1035)

1 ro  et (0.01 sec)
```

Query 2:

```
mysql> DROP INDEX idx_userid_appliedat;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1
mysql> DROP INDEX idx_appliedat ON Application;
ERROR 1091 (42000): Can't DROP 'idx_appliedat'; check that column/key exists
mysql> DROP INDEX idx_userid_appliedat ON Application;
ERROR 1553 (HY000): Cannot drop index 'idx_userid_appliedat': needed in a foreign key constraint
mysql> DROP INDEX idx_userid_appliedat ON Application;
ERROR 1553 (HY000): Cannot drop index 'idx_userid_appliedat': needed in a foreign key constraint
mysql> CREATE INDEX idx_userid_appliedat ON Application(UserID, AppliedAt);
ERROR 1061 (42000): Duplicate key name 'idx_userid_appliedat'
mysql> DROP INDEX idx_appliedat ON Application;
ERROR 1091 (42000): Can't DROP 'idx_appliedat'; check that column/key exists
mysql> DROP INDEX idx_appliedat ON Application;
ERROR 1091 (42000): Can't DROP 'idx_appliedat'; check that column/key exists
mysql> DROP INDEX idx_jobid_app ON Application;
ERROR 1091 (42000): Can't DROP 'idx_jobid_app'; check that column/key exists
mysql> DROP INDEX idx_jobid_job ON Job;
ERROR 1091 (42000): Can't DROP 'idx_jobid_job'; check that column/key exists
mysql> DROP INDEX idx_salary_job ON Job;
ERROR 1091 (42000): Can't DROP 'idx_salary_job'; check that column/key exists
mysql> EXPLAIN ANALYZE
    -> SELECT
    ->     u.UserID,
    ->     u.Name,
    ->     j.JobID,
    ->     j.Title,
    ->     a.AppliedAt
    -> FROM
    ->     User u
    -> JOIN
    ->     Application a ON u.UserID = a.UserID
    -> JOIN
    ->     Job j ON a.JobID = j.JobID
    -> WHERE
    ->     a.AppliedAt = (
    ->         SELECT MAX(a2.AppliedAt)
    ->         FROM Application a2
    ->         WHERE a2.UserID = u.UserID
    ->     );

| EXPLAIN

| -> Nested loop inner join  (cost=771 rows=934) (actual time=0.428..14.9 rows=1000 loops=1)
    -> Nested loop inner join  (cost=445 rows=934) (actual time=0.376..12.9 rows=1000 loops=1)
        -> Table scan on u  (cost=118 rows=934) (actual time=0.191..1.21 rows=1008 loops=1)
        -> Filter: (a.AppliedAt = (select #2))  (cost=0.25 rows=1) (actual time=0.011..0.0115 rows=0.992 loops=1008)
            -> Index lookup on a using idx_userid_appliedat (UserID = u.UserID, AppliedAt = (select #2))  (cost=0.25 rows=1) (actual time=0.00835..0.00872 rows=0.992 loops=1008)
            -> Select #2 (subquery in condition; dependent)
                -> Aggregate: max(a2.AppliedAt)  (cost=0.967 rows=1) (actual time=0.00222..0.00224 rows=1 loops=3008)
                    -> Covering index lookup on a2 using idx_userid_appliedat (UserID = u.UserID)  (cost=0.729 rows=1.03) (actual time=0.00158..0.00195 rows=1.03 loops=3008)
    -> Single-row index lookup on j using PRIMARY (JobID = a.JobID)  (cost=0.25 rows=1) (actual time=0.0019..0.00192 rows=1 loops=1000)

1 row in set, 1 warning (0.02 sec)
```

Query 3:

```
1 row in set, 1 warning (0.02 sec)

mysql> EXPLAIN ANALYZE
    -> SELECT
    ->     u.UserID,
    ->     u.Name,
    ->     j.Company,
    ->     COUNT(DISTINCT j.JobID) AS NumJobsAtCompany
    -> FROM
    ->     User u
    -> JOIN
    ->     Application a ON u.UserID = a.UserID
    -> JOIN
    ->     Job j ON a.JobID = j.JobID
    -> GROUP BY
    ->     u.UserID, u.Name, j.Company;
+--------------------------------------------------------------------------------
| EXPLAIN
+--------------------------------------------------------------------------------
| -> Group aggregate: count(distinct job.JobID)  (actual time=7.76..8.28 rows=1033 loops=1)
    -> Sort: u.UserID, u.`Name`, j.Company  (actual time=7.71..7.78 rows=1035 loops=1)
        -> Stream results  (cost=794 rows=967) (actual time=0.285..6.8 rows=1035 loops=1)
            -> Nested loop inner join  (cost=794 rows=967) (actual time=0.278..6.44 rows=1035 loops=1)
                -> Nested loop inner join  (cost=456 rows=967) (actual time=0.244..4.43 rows=1035 loops=1)
                    -> Table scan on u  (cost=118 rows=934) (actual time=0.16..1.02 rows=1008 loops=1)
                    -> Index lookup on a using idx_userid_appliedat (UserID = u.UserID)  (cost=0.259 rows=1.03) (actual time=0.00279..0.00321 rows=1.03 loops=1008)
                -> Single-row index lookup on j using PRIMARY (JobID = a.JobID)  (cost=0.25 rows=1) (actual time=0.00153..0.00182 rows=1 loops=1035)
 |
+--------------------------------------------------------------------------------
1 ro   et (0.01 sec)
```

Query 4:

```
--------------------------------------------------------------------------------
1 row in set (0.01 sec)

mysql> EXPLAIN ANALYZE
    -> SELECT
    ->     DISTINCT u.UserID,
    ->     u.Name,
    ->     u.Email
    -> FROM
    ->     User u
    -> JOIN
    ->     Application a ON u.UserID = a.UserID
    -> WHERE
    ->     u.UserID NOT IN (
    ->         SELECT a.UserID
    ->         FROM Application a
    ->         JOIN Job j ON a.JobID = j.JobID
    ->         WHERE j.Salary = (SELECT MAX(Salary) FROM Job)
    ->     );
+--------------------------------------------------------------------------------
--------------------------+
| EXPLAIN
                                                                                     |
+--------------------------------------------------------------------------------
--------------------------+
| -> Table scan on <temporary>  (cost=354519..367912 rows=1.07e+6) (actual time=3.63..3.73 rows=1000 loops=1)
    -> Temporary table with deduplication  (cost=354519..354519 rows=1.07e+6) (actual time=3.63..3.63 rows=1000 loops=1)
        -> Nested loop antijoin  (cost=107693 rows=1.07e+6) (actual time=0.47..2.81 rows=1035 loops=1)
            -> Nested loop inner join  (cost=467 rows=1035) (actual time=0.112..2.07 rows=1035 loops=1)
                -> Covering index scan on a using idx_userid_appliedat  (cost=105 rows=1035) (actual time=0.0878..0.267 rows=1035 loops=1)
                -> Single-row index lookup on u using PRIMARY (UserID = a.UserID)  (cost=0.25 rows=1) (actual time=0.0016..0.00163 rows=1 loops=1035)
            -> Single-row index lookup on <subquery2> using <auto_distinct_key> (UserID = a.UserID)  (cost=704..704 rows=1) (actual time=619e-6..619e-6 rows=0 loops=1035)
                -> Materialize with deduplication  (cost=704..704 rows=1035) (actual time=0.353..0.353 rows=0 loops=1)
                    -> Filter: (a.UserID is not null)  (cost=465 rows=1035) (actual time=0.35..0.35 rows=0 loops=1)
                        -> Nested loop inner join  (cost=465 rows=1035) (actual time=0.35..0.35 rows=0 loops=1)
                            -> Filter: (j.Salary = (select #3))  (cost=103 rows=1000) (actual time=0.35..0.35 rows=0 loops=1)
                                -> Table scan on j  (cost=103 rows=1000) (actual time=0.0605..0.313 rows=1000 loops=1)
                                -> Select #3 (subquery in condition; run only once)
                                    -> Aggregate: max(job.Salary)  (cost=334 rows=1) (never executed)
                                        -> Table scan on Job  (cost=103 rows=1000) (never executed)
                            -> Index lookup on a using JobID (JobID = j.JobID)  (cost=0.259 rows=1.03) (never executed)
 |
+--------------------------------------------------------------------------------
--------------------------+
1 row in set (0.01 sec)
```

Query 5:

```
1 row in set (0.01 sec)

mysql> EXPLAIN ANALYZE
    -> SELECT
    ->     u.UserID,
    ->     COUNT(a.ApplicationID) AS TotalApplications
    -> FROM
    ->     User u
    -> JOIN
    ->     Application a ON u.UserID = a.UserID
    -> GROUP BY
    ->     u.UserID
    -> HAVING
    ->     COUNT(a.ApplicationID) >= (
    ->         SELECT AVG(app_count)
    ->         FROM (
    ->             SELECT COUNT(*) AS app_count
    ->             FROM Application
    ->             GROUP BY UserID
    ->         ) AS sub
    ->     );
+------------------------------------------------------------------------------------------------------------
| EXPLAIN
                                                                                              |
+------------------------------------------------------------------------------------------------------------
| -> Filter: (`count(a.ApplicationID)` >= (select #2))  (actual time=2.73..2.89 rows=25 loops=1)
    -> Table scan on <temporary>  (actual time=2.12..2.2 rows=1000 loops=1)
        -> Aggregate using temporary table  (actual time=2.12..2.12 rows=1000 loops=1)
            -> Nested loop inner join  (cost=467 rows=1035) (actual time=0.107..1.75 rows=1035 loops=1)
                -> Covering index scan on a using idx_userid_appliedat  (cost=105 rows=1035) (actual time=0.0835..0.254 rows=1035 loops=1)
                -> Single-row covering index lookup on u using PRIMARY (UserID = a.UserID)  (cost=0.25 rows=1) (actual time=0.00131..0.00133 rows=1 loops=1035)
    -> Select #2 (subquery in condition; run only once)
        -> Aggregate: avg(sub.app_count)  (cost=819..819 rows=1) (actual time=0.57..0.57 rows=1 loops=1)
            -> Table scan on sub  (cost=574..589 rows=1000) (actual time=0.431..0.503 rows=1000 loops=1)
                -> Materialize  (cost=574..574 rows=1000) (actual time=0.43..0.43 rows=1000 loops=1)
                    -> Group aggregate: count(0)  (cost=343 rows=1000) (actual time=0.0792..0.31 rows=1000 loops=1)
                        -> Covering index scan on Application using idx_userid_appliedat  (cost=105 rows=1035) (actual time=0.0734..0.209 rows=1035 loops=1)
 |
+------------------------------------------------------------------------------------------------------------
1 ro    et (0.00 sec)
```

Design C:

CREATE INDEX idx_userid_app ON Application(UserID);

CREATE INDEX idx_jobid_app ON Application(JobID);

CREATE INDEX idx_jobid_job ON Job(JobID);

Query 1:

```
mysql> DROP INDEX idx_appliedat ON Application;
ERROR 1091 (42000): Can't DROP 'idx_appliedat'; check that column/key exists
mysql> CREATE INDEX idx_userid_app ON Application(UserID);
CREATE INDEX idx_jobid_app ON Application(JobID);
CREATE INDEX idx_jobid_job ON Job(JobID);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> CREATE INDEX idx_jobid_app ON Application(JobID);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> CREATE INDEX idx_jobid_job ON Job(JobID);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE
    -> SELECT
    ->     a.ApplicationID,
    ->     u.Name,
    ->     j.Title,
    ->     a.AppliedAt
    -> FROM
    ->     Application a
    -> JOIN
    ->     User u ON a.UserID = u.UserID
    -> JOIN
    ->     Job j ON a.JobID = j.JobID
    -> WHERE
    ->     a.ApplicationID IN (
    ->         SELECT a2.ApplicationID
    ->         FROM Application a2
    ->         WHERE a2.AppliedAt >= NOW() - INTERVAL 45 DAY
    ->     )
    -> ORDER BY
    ->     a.AppliedAt DESC;
+--------------------------------------------------------------------------------+
| EXPLAIN                                                                        |
+--------------------------------------------------------------------------------+
| -> Sort: a.AppliedAt DESC  (actual time=9.97..10.1 rows=1035 loops=1)
    -> Stream results  (cost=398 rows=345) (actual time=0.257..9.5 rows=1035 loops=1)
        -> Nested loop inner join  (cost=398 rows=345) (actual time=0.249..8.86 rows=1035 loops=1)
            -> Nested loop inner join  (cost=277 rows=345) (actual time=0.193..6.23 rows=1035 loops=1)
                -> Nested loop inner join  (cost=156 rows=345) (actual time=0.148..3.18 rows=1035 loops=1)
                    -> Filter: (a2.AppliedAt >= <cache>((now() - interval 45 day)))  (cost=35.7 rows=345) (actual time=0.0993..0.645 rows=1035 loops=1)
                        -> Covering index scan on a2 using idx_userid_appliedat  (cost=35.7 rows=1035) (actual time=0.0772..0.502 rows=1035 loops=1)
                    -> Single-row index lookup on a using PRIMARY (ApplicationID = a2.ApplicationID)  (cost=0.25 rows=1) (actual time=0.00223..0.00227 rows=1 loops=1035)
                -> Single-row index lookup on u using PRIMARY (UserID = a.UserID)  (cost=0.25 rows=1) (actual time=0.00273..0.00276 rows=1 loops=1035)
            -> Single-row index lookup on j using PRIMARY (JobID = a.JobID)  (cost=0.25 rows=1) (actual time=0.00232..0.00235 rows=1 loops=1035)
 |
+--------------------------------------------------------------------------------+
1 ro[illegible]et (0.02 sec)
```

Query 2:

```
mysql> EXPLAIN ANALYZE
    -> SELECT
    ->     u.UserID,
    ->     u.Name,
    ->     j.JobID,
    ->     j.Title,
    ->     a.AppliedAt
    -> FROM
    ->     User u
    -> JOIN
    ->     Application a ON u.UserID = a.UserID
    -> JOIN
    ->     Job j ON a.JobID = j.JobID
    -> WHERE
    ->     a.AppliedAt = (
    ->         SELECT MAX(a2.AppliedAt)
    ->         FROM Application a2
    ->         WHERE a2.UserID = u.UserID
    ->     );
| EXPLAIN
| -> Nested loop inner join  (cost=771 rows=934) (actual time=0.235..19.3 rows=1000 loops=1)
    -> Nested loop inner join  (cost=445 rows=934) (actual time=0.224..17 rows=1000 loops=1)
        -> Table scan on u  (cost=118 rows=934) (actual time=0.112..1.16 rows=1008 loops=1)
        -> Filter: (a.AppliedAt = (select #2))  (cost=0.25 rows=1) (actual time=0.0151..0.0156 rows=0.992 loops=1008)
            -> Index lookup on a using idx_userid_appliedat (UserID = u.UserID, AppliedAt = (select #2))  (cost=0.25 rows=1) (actual time=0.0108..0.0113 rows=0.992 loops=1008)
            -> Select #2 (subquery in condition; dependent)
                -> Aggregate: max(a2.AppliedAt)  (cost=0.601 rows=1) (actual time=0.00327..0.0033 rows=1 loops=3008)
                    -> Index lookup on a2 using idx_userid_app (UserID = u.UserID)  (cost=0.362 rows=1.03) (actual time=0.00254..0.00293 rows=1.03 loops=3008)
    -> Single-row index lookup on j using PRIMARY (JobID = a.JobID)  (cost=0.25 rows=1) (actual time=0.00194..0.00196 rows=1 loops=1000)
 |
1 ro   et, 1 warning (0.03 sec)
```

Query 3:

```
mysql> EXPLAIN ANALYZE
    -> SELECT
    ->     u.UserID,
    ->     u.Name,
    ->     j.Company,
    ->     COUNT(DISTINCT j.JobID) AS NumJobsAtCompany
    -> FROM
    ->     User u
    -> JOIN
    ->     Application a ON u.UserID = a.UserID
    -> JOIN
    ->     Job j ON a.JobID = j.JobID
    -> GROUP BY
    ->     u.UserID, u.Name, j.Company;
| EXPLAIN
| -> Group aggregate: count(distinct job.JobID)  (actual time=7.4..7.95 rows=1033 loops=1)
    -> Sort: u.UserID, u.`Name`, j.Company  (actual time=7.38..7.46 rows=1035 loops=1)
        -> Stream results  (cost=794 rows=967) (actual time=0.145..6.73 rows=1035 loops=1)
            -> Nested loop inner join  (cost=794 rows=967) (actual time=0.139..6.39 rows=1035 loops=1)
                -> Nested loop inner join  (cost=456 rows=967) (actual time=0.128..4.15 rows=1035 loops=1)
                    -> Table scan on u  (cost=118 rows=934) (actual time=0.0844..0.98 rows=1008 loops=1)
                    -> Index lookup on a using idx_userid_appliedat (UserID = u.UserID)  (cost=0.259 rows=1.03) (actual time=0.00238..0.00302 rows=1.03 loops=1008)
                -> Single-row index lookup on j using PRIMARY (JobID = a.JobID)  (cost=0.25 rows=1) (actual time=0.002..0.00204 rows=1 loops=1035)
 |
1 ro   et (0.01 sec)
Calculat
```

Query 4:

```
mysql> EXPLAIN ANALYZE
    -> SELECT
    ->     DISTINCT u.UserID,
    ->     u.Name,
    ->     u.Email
    -> FROM
    ->     User u
    -> JOIN
    ->     Application a ON u.UserID = a.UserID
    -> WHERE
    ->     u.UserID NOT IN (
    ->         SELECT a.UserID
    ->         FROM Application a
    ->         JOIN Job j ON a.JobID = j.JobID
    ->         WHERE j.Salary = (SELECT MAX(Salary) FROM Job)
    ->     );
+----------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                      |
+----------------------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (cost=416522..428608 rows=966690) (actual time=5.81..5.99 rows=1000 loops=1)
    -> Temporary table with deduplication  (cost=416522..416522 rows=966690) (actual time=5.81..5.81 rows=1000 loops=1)
        -> Nested loop inner join  (cost=193783 rows=966690) (actual time=0.59..3.85 rows=1000 loops=1)
            -> Nested loop antijoin  (cost=96880 rows=966690) (actual time=0.565..2.07 rows=1008 loops=1)
                -> Table scan on u  (cost=118 rows=934) (actual time=0.135..0.992 rows=1008 loops=1)
                -> Single-row index lookup on <subquery2> using <auto_distinct_key> (UserID = u.UserID)  (cost=704..704 rows=1) (actual time=915e-6..915e-6 rows=0 loops=1008)
                    -> Materialize with deduplication  (cost=704..704 rows=1035) (actual time=0.419..0.419 rows=0 loops=1)
                        -> Filter: (a.UserID is not null)  (cost=465 rows=1035) (actual time=0.416..0.416 rows=0 loops=1)
                            -> Nested loop inner join  (cost=465 rows=1035) (actual time=0.416..0.416 rows=0 loops=1)
                                -> Filter: (j.Salary = (select #3))  (cost=103 rows=1000) (actual time=0.416..0.416 rows=0 loops=1)
                                    -> Table scan on j  (cost=103 rows=1000) (actual time=0.0896..0.377 rows=1000 loops=1)
                                    -> Select #3 (subquery in condition; run only once)
                                        -> Aggregate: max(job.Salary)  (cost=334 rows=1) (never executed)
                                            -> Table scan on Job  (cost=103 rows=1000) (never executed)
                                -> Index lookup on a using idx_jobid_app (JobID = j.JobID)  (cost=0.259 rows=1.03) (never executed)
            -> Limit: 1 row(s)  (cost=0.25 rows=1) (actual time=0.00161..0.00163 rows=0.992 loops=1008)
                -> Covering index lookup on a using idx_userid_app (UserID = u.UserID)  (cost=0.25 rows=1.03) (actual time=0.00152..0.00152 rows=0.992 loops=1008)
 |
+----------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)
```

Query 5:

```
mysql> EXPLAIN ANALYZE
    -> SELECT
    ->     u.UserID,
    ->     COUNT(a.ApplicationID) AS TotalApplications
    -> FROM
    ->     User u
    -> JOIN
    ->     Application a ON u.UserID = a.UserID
    -> GROUP BY
    ->     u.UserID
    -> HAVING
    ->     COUNT(a.ApplicationID) >= (
    ->         SELECT AVG(app_count)
    ->         FROM (
    ->             SELECT COUNT(*) AS app_count
    ->             FROM Application
    ->             GROUP BY UserID
    ->         ) AS sub
    ->     );
+--------------------------------------------------------------------------------+
| EXPLAIN                                                                        |
+--------------------------------------------------------------------------------+
| -> Filter: (count(a.ApplicationID) >= (select #2))  (cost=671 rows=927) (actual time=1.27..3.62 rows=25 loops=1)
    -> Group aggregate: count(a.ApplicationID), count(a.ApplicationID)  (cost=671 rows=927) (actual time=0.524..2.71 rows=1000 loops=1)
        -> Nested loop inner join  (cost=448 rows=967) (actual time=0.504..2.52 rows=1035 loops=1)
            -> Covering index scan on u using PRIMARY  (cost=118 rows=934) (actual time=0.465..0.687 rows=1008 loops=1)
            -> Covering index lookup on a using idx_userid_app (UserID = u.UserID)  (cost=0.25 rows=1.03) (actual time=0.00117..0.0016 rows=1.03 loops=1008)
    -> Select #2 (subquery in condition; run only once)
        -> Aggregate: avg(sub.app_count)  (cost=819..819 rows=1) (actual time=0.709..0.71 rows=1 loops=1)
            -> Table scan on sub  (cost=574..589 rows=1000) (actual time=0.528..0.623 rows=1000 loops=1)
                -> Materialize  (cost=574..574 rows=1000) (actual time=0.526..0.526 rows=1000 loops=1)
                    -> Group aggregate: count(0)  (cost=343 rows=1000) (actual time=0.0665..0.369 rows=1000 loops=1)
                        -> Covering index scan on Application using idx_userid_app  (cost=105 rows=1035) (actual time=0.0643..0.241 rows=1035 loops=1)
 |
+--------------------------------------------------------------------------------+
1 ro[...]et (0.01 sec)
```