1. Indexing: Ensure that appropriate indexes are created on the columns used in the join and filter conditions. Proper indexing can significantly speed up query execution.

2. Limit the Use of Wildcards (%): The use of wildcards at the beginning of LIKE clauses (e.g., '%キャビンアテンダント%') can be resource-intensive. If possible, try to avoid leading wildcards or use full-text search mechanisms for better performance.

3. Limit the Number of Joins: The query involves several LEFT JOINs, which can affect the execution time. Consider whether all the joins are necessary or if some can be optimized. Only join necessary table(s).

4. Table Partitioning: If the tables are very large, consider partitioning them based on specific criteria (e.g., date ranges) to improve query performance.

5. Materialized Views: Try materialized views to store the results of certain complex joins or aggregations. This also saves time for future use.

6. Query Plan Analysis: Analyze the query execution plan to identify any performance bottlenecks. Use EXPLAIN statement (or its equivalent in your database system) to analyze the query plan that affect the execution time and optimize accordingly.

7. Caching: Implement caching mechanisms to store the results of frequently executed queries and reduce database load.

8. Query Pagination: For large result sets, consider using pagination to fetch data in smaller batches, rather than retrieving all records at once. Can be use LIMIT in the query.

9. Optimize Server and Database Configuration: Ensure that your database server is properly configured, with sufficient resources allocated to handle the query load. The larger the ram the better for optimization.

10. Application-Level Optimization: Review the application code that generates this query and optimize it, if possible, try to reduce the complexity of the generated SQL.

Remember that the effectiveness of these optimization tips may vary depending on the specific data, database schema, and workload. It's essential to thoroughly test and benchmark any changes made to ensure they indeed improve query performance. Additionally, regular database maintenance, such as index rebuilding and statistics updates, can also contribute to better performance over time.

Below are the improved query.

```
SELECT Jobs.id AS `Jobs__id`,
Jobs.name AS `Jobs__name`,
Jobs.media_id AS `Jobs__media_id`,
Jobs.job_category_id AS `Jobs__job_category_id`,
Jobs.job_type_id AS `Jobs__job_type_id`,
Jobs.description AS `Jobs__description`,
Jobs.detail AS `Jobs__detail`,
Jobs.business_skill AS `Jobs__business_skill`,
Jobs.knowledge AS `Jobs__knowledge`,
Jobs.location AS `Jobs__location`,
Jobs.activity AS `Jobs__activity`,
Jobs.academic_degree_doctor AS `Jobs__academic_degree_doctor`,
Jobs.academic_degree_master AS `Jobs__academic_degree_master`,
Jobs.academic_degree_professional AS `Jobs__academic_degree_professional`,
Jobs.academic_degree_bachelor AS `Jobs__academic_degree_bachelor`,
Jobs.salary_statistic_group AS `Jobs__salary_statistic_group`,
Jobs.salary_range_first_year AS `Jobs__salary_range_first_year`,
Jobs.salary_range_average AS `Jobs__salary_range_average`,
Jobs.salary_range_remarks AS `Jobs__salary_range_remarks`,
Jobs.restriction AS `Jobs__restriction`,
Jobs.estimated_total_workers AS `Jobs__estimated_total_workers`,
Jobs.remarks AS `Jobs__remarks`,
Jobs.url AS `Jobs__url`,
Jobs.seo_description AS `Jobs__seo_description`,
Jobs.seo_keywords AS `Jobs__seo_keywords`,
Jobs.sort_order AS `Jobs__sort_order`,
Jobs.publish_status AS `Jobs__publish_status`,
Jobs.version AS `Jobs__version`,
Jobs.created_by AS `Jobs__created_by`,
Jobs.created AS `Jobs__created`,
Jobs.modified AS `Jobs__modified`,
Jobs.deleted AS `Jobs__deleted`,
JobCategories.id AS `JobCategories__id`,
JobCategories.name AS `JobCategories__name`,
JobCategories.sort_order AS `JobCategories__sort_order`,
JobCategories.created_by AS `JobCategories__created_by`,
JobCategories.created AS `JobCategories__created`,
JobCategories.modified AS `JobCategories__modified`,
JobCategories.deleted AS `JobCategories__deleted`,
JobTypes.id AS `JobTypes__id`,
JobTypes.name AS `JobTypes__name`,
JobTypes.job_category_id AS `JobTypes__job_category_id`,
JobTypes.sort_order AS `JobTypes__sort_order`,
JobTypes.created_by AS `JobTypes__created_by`,
JobTypes.created AS `JobTypes__created`,
JobTypes.modified AS `JobTypes__modified`,
JobTypes.deleted AS `JobTypes__deleted`
FROM jobs Jobs
LEFT JOIN jobs_personalities JobsPersonalities
ON Jobs.id = (JobsPersonalities.job_id)
LEFT JOIN personalities Personalities
ON (Personalities.id = (JobsPersonalities.personality_id)
AND (Personalities.deleted) IS NULL)
LEFT JOIN jobs_practical_skills JobsPracticalSkills
ON Jobs.id = (JobsPracticalSkills.job_id)
LEFT JOIN practical_skills PracticalSkills
ON (PracticalSkills.id = (JobsPracticalSkills.practical_skill_id)
AND (PracticalSkills.deleted) IS NULL)
LEFT JOIN jobs_basic_abilities JobsBasicAbilities
ON Jobs.id = (JobsBasicAbilities.job_id)
LEFT JOIN basic_abilities BasicAbilities
ON (BasicAbilities.id = (JobsBasicAbilities.basic_ability_id)
AND (BasicAbilities.deleted) IS NULL)
LEFT JOIN jobs_tools JobsTools
ON Jobs.id = (JobsTools.job_id)
LEFT JOIN affiliates Tools
ON (Tools.type = 1
AND Tools.id = (JobsTools.affiliate_id)
AND (Tools.deleted) IS NULL)
LEFT JOIN jobs_career_paths JobsCareerPaths
ON Jobs.id = (JobsCareerPaths.job_id)
LEFT JOIN affiliates CareerPaths
ON (CareerPaths.type = 3
AND CareerPaths.id = (JobsCareerPaths.affiliate_id)
AND (CareerPaths.deleted) IS NULL)
LEFT JOIN jobs_rec_qualifications JobsRecQualifications
ON Jobs.id = (JobsRecQualifications.job_id)
LEFT JOIN affiliates RecQualifications
ON (RecQualifications.type = 2
AND RecQualifications.id = (JobsRecQualifications.affiliate_id)
AND (RecQualifications.deleted) IS NULL)
LEFT JOIN jobs_req_qualifications JobsReqQualifications
ON Jobs.id = (JobsReqQualifications.job_id)
LEFT JOIN affiliates ReqQualifications
ON (ReqQualifications.type = 2
AND ReqQualifications.id = (JobsReqQualifications.affiliate_id)
AND (ReqQualifications.deleted) IS NULL)
INNER JOIN job_categories JobCategories
ON (JobCategories.id = (Jobs.job_category_id)
AND (JobCategories.deleted) IS NULL)
```

```
INNER JOIN job_types JobTypes
ON (JobTypes.id = (Jobs.job_type_id)
AND (JobTypes.deleted) IS NULL)
WHERE ((JobCategories.name LIKE '%キャビンアテンダント%')
AND publish_status = 1
AND (Jobs.deleted) IS NULL)
GROUP BY Jobs.id
ORDER BY Jobs.sort_order desc,
Jobs.id DESC LIMIT 50 OFFSET 0
```