**Задачи по SQL**

**Удалить дубли из таблицы**

Имеется таблица TBL состоящая из одного поля fld int

|  |
| --- |
| fld |
| 1 |
| 2 |
| 3 |
| 3 |
| 4 |
| 5 |

Необходимо удалить дублирующие по значению записи из таблицы TBL, оставив только уникальные значения, т. е. результат в таблице TBL должен выглядеть следующим образом:

|  |
| --- |
| Fld |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |

**Решение**:

DELETE t3 FROM t3

INNER JOIN (SELECT Min(`ld`) minid, fld FROM t3

GROUP BY fld HAVING Count(1) > 1) AS duplicatesTable

ON ( duplicatesTable.fld = t3.fld

AND duplicatesTable.minid <> `ld` )

**Рассчитать сумму в рублях**

Имеется две таблицы: Проводки (**CARRY**)

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | Код валюты | Сумма в валюте | Сумма в рублях |
| 05.02.2017 | USD | 100 |  |
| 06.02.2017 | USD | 200 |  |
| 07.02.2017 | USD | 300 |  |

Курсы валют (**CURRENCY\_RATE**)

|  |  |  |
| --- | --- | --- |
| Дата | Код валюты | Курс |
| 05.01.2017 | USD | 61 |
| 08.01.2017 | USD | 62 |
| 09.01.2017 | USD | 63 |

Необходимо для каждой проводки в таблице CARRY рассчитать соответствующую Сумму в рублях (Сумма в валюте \* Ближайший по дате курс валюты)

**Решение**:

SELECT c.`Дата`,

c.`Код валюты`,

c.`Сумма в валюте`,

c.`Сумма в валюте` \* (SELECT cr1.`Курс` FROM `currency\_rate` cr1 WHERE cr1.`Дата` = (SELECT MAX(cr2.`Дата`) FROM `currency\_rate` cr2 WHERE cr2.`Дата`<= c.`Дата`)) AS `сумма в рублях`

FROM `carry` c

SELECT `Дата`,

`Код валюты`,

`Сумма в валюте`,

`Сумма в валюте` \* (SELECT `Курс` FROM `currency\_rate` WHERE `Дата` = (SELECT MAX(`Дата`) FROM `currency\_rate` WHERE `currency\_rate`.`Дата` <= `carry`.`Дата`)) AS `сумма в рублях`

FROM carry

**Анализ по иерархии**

Имеется таблица Сотрудники (**Employee**)

|  |  |  |
| --- | --- | --- |
| Код сотрудника | Код непосредственного руководителя | Оклад |
| 1 |  | 100 |
| 2 | 1 | 200 |
| 3 | 2 | 150 |

Необходимо вывести все сочетания сотрудников, когда оклад руководителя (необязательно непосредственного) меньше, чем оклад сотрудника.

В данном примере это сочетания 1 - 2, 1 - 3

**Решение**:  CREATE TEMPORARY TABLE `tmp\_p`

SELECT \* FROM `employee` WHERE `Код непосредственного руководителя` IS NOT NULL;

**Построение диапазонов**

Имеется таблица TBL(fld int)

|  |
| --- |
| Fld |
| 1 |
| 2 |
| 5 |
| 6 |
| 7 |
| 9 |

Необходимо в одном запросе вывести все диапазоны, внутри которых значение поля **fld** непрерывно (по значению)

В данном примере результат выглядит следующим образом

|  |  |
| --- | --- |
| fld\_from | fld\_to |
| 1 | 2 |
| 5 | 7 |
| 9 | 9 |

**Решение**:

SELECT MIN(`fld`) AS fld\_from, MAX(`fld`) AS fld\_to FROM

(SELECT `fld`, (SELECT MIN(B.`fld`) FROM `tbl` AS B

WHERE B.`fld` >= A.`fld`

AND NOT EXISTS

(SELECT \* FROM `tbl` AS C

WHERE C.`fld` = B.`fld` + 1)) AS x

FROM `tbl` AS A) AS D

GROUP BY x;

**Бо́льшие зарплаты**

В БД есть таблица Salaries, в которой у каждого сотрудника есть идентификатор, а также столбец для идентификатора отдела. Напишите SQL-запрос, чтобы найти сотрудников, которые получают три самые высокие зарплаты в каждом отделе.

**Salaries**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Salary | DepartmentId |
| 1 | Petr | 85000 | 1 |
| 2 | Ivan | 80000 | 2 |
| 3 | Alex | 60000 | 2 |
| 4 | Den | 90000 | 1 |
| 5 | Bob | 69000 | 1 |
| 6 | Kir | 85000 | 1 |
| 7 | Mike | 76000 | 1 |

**Department**

|  |  |
| --- | --- |
| DepartmentId | DepartmentName |
| 1 | IT |
| 2 | Sales |

Запрос должен возвращать следующие строки (порядок не имеет значения):

|  |  |  |
| --- | --- | --- |
| Department | Employee | Salary |
| IT | Den | 90000 |
| IT | Petr | 85000 |
| IT | Kir | 85000 |
| IT | Mike | 76000 |
| Sales | Ivan | 80000 |
| Sales | Alex | 60000 |

**Решение**:

SELECT `department`.`DepartmentName`, s.`Name`, s.`Salary`

FROM `salaries` s JOIN `department`

ON s.`DepartmentId` = `department`.`DepartmentId`

WHERE (SELECT COUNT(DISTINCT sal.`Salary`) FROM `salaries` sal WHERE sal.Salary > s.`Salary` AND sal.`DepartmentId` = s.`DepartmentId`) < 3

ORDER BY `department`.`DepartmentName` ASC, `s`.`Salary` DESC

**Вставка задним числом**

Есть таблица с двумя полями **Id** и **Timestamp**, где

**Id**- возрастающая последовательность, каждая вставка новой записи в таблицу приводит к генерации ID(n)=ID(n-1) + 1

**Timestamp** – временная метка, в стандартном процессе текущее время, при вставке задним числом может принимать любые значения меньше максимума времени всех предыдущих записей

Вставка задним числом – операция вставки записи в таблицу, при которой

**ID(n) > ID(n-1)**

**Timestamp(n) < max(timestamp(1):timestamp(n-1))**

Пример таблицы:

|  |  |
| --- | --- |
| id | timestamp |
| 1 | 2016.09.11 |
| 2 | 2016.09.12 |
| 3 | 2016.09.13 |
| 4 | 2016.09.14 |
| 5 | 2016.09.09 |
| 6 | 2016.09.08 |
| 7 | 2016.09.15 |

Вставка задним числом - записи 5 и 6

Написать код, который будет возвращать список всех id, подходящих под определение вставки задним числом.

**Решение**:

declare @id int

declare @maxid int

set @id = 1

select @maxid = max(`ld`) from `times`

create temporary table `IDSeq`(id int)

while @id < @maxid

begin

insert into `IDSeq` values(@id)

set @id = @id + 1

end

select s.id from `IDSeq` s left join `times` t

on s.id = t.id

where t.id is null

drop table `IDSeq`

SELECT

id1+1 AS 'FROM',

id3-1 AS 'TO'

FROM (

SELECT

t1.id AS id1,

t2.id AS id2,

(

SELECT t3.id FROM (SELECT times.id FROM times) t3

WHERE t3.id > t1.id

ORDER BY t1.ld

LIMIT 1

) AS id3 FROM (SELECT times.id FROM times) t1

LEFT JOIN (SELECT times.id FROM times) t2 ON t2.id = t1.id + 1) A

WHERE id1 IS NOT NULL AND id2 IS NULL AND id3 IS NOT NULL

CREATE TABLE test12 (

id INT NOT NULL PRIMARY KEY

);

INSERT INTO test12 (test12.id) SELECT `times`.`ld` FROM `times`;

SELECT

id1+1 AS 'FROM',

id3-1 AS 'TO'

FROM (

SELECT

t1.id AS id1,

t2.id AS id2,

(

SELECT id FROM test12 t3

WHERE t3.id > t1.id

ORDER BY id

LIMIT 1

) AS id3 FROM test12 t1

LEFT JOIN test12 t2 ON t2.id = t1.id + 1) A

WHERE id1 IS NOT NULL AND id2 IS NULL AND id3 IS NOT NULL

CREATE TEMPORARY TABLE `tmp\_test12` (

id INT NOT NULL PRIMARY KEY

);

INSERT INTO `tmp\_test12` (`tmp\_test12`.id) SELECT `times`.`ld` FROM `times`;

SELECT

id1+1 AS 'FROM',

id3-1 AS 'TO'

FROM (

SELECT

t1.id AS id1,

t2.id AS id2,

(

SELECT id FROM `tmp\_test12` t3

WHERE t3.id > t1.id

ORDER BY id

LIMIT 1

) AS id3 FROM `tmp\_test12` t1

LEFT JOIN `tmp\_test12` t2 ON t2.id = t1.id + 1) A

WHERE id1 IS NOT NULL AND id2 IS NULL AND id3 IS NOT NULL

SELECT id+1 as start\_interval, next\_id-1 as finish\_interval FROM

(SELECT id,

LEAD(id)OVER(ORDER BY Id)as next\_id FROM

(SELECT 0 Id UNION ALL SELECT Id FROM times)T)T