

```
1 -- This is a list of the SQL queries we executed in class
2
3
4 -- SQL queries using one relation
5
6 -- What is the population of the US?
7 SELECT Population
8 FROM Country
9 WHERE Code = 'USA';
10
11 -- Which countries gained independence after 1989?
12 SELECT Name, IndepYear
13 FROM Country
14 WHERE IndepYear >= '1990';
15
16 -- Return all the attributes for cities with population over 1 million in the US
17 SELECT *
18 FROM City
19 WHERE Population >= '1000000' AND CountryCode = 'USA';
20
21 -- This query returns only the name of these cities, and renames the attribute
22 -- Name to LargeUSACity
23 SELECT Name AS LargeUSACity
24 FROM City
25 WHERE Population >= '1000000' AND CountryCode = 'USA';
26
27 -- Cities with population over 5 million (changing the unit to millions)
28 SELECT Name, (Population / 1000000) AS PopulationInMillion
29 FROM City
30 WHERE Population >= '5000000';
31
32 -- If we want to get a float value, and not the rounded integer value
33 SELECT Name, ((Population * 1.0) / 1000000) AS PopulationInMillion
34 FROM City
35 WHERE Population >= '5000000';
36
37 -- Find the countries that have a form of goverment related to monarchy
38 SELECT Name, GovernmentForm
39 FROM Country
40 WHERE GovernmentForm LIKE '%Monarchy%';
41
42 -- Find all forms of goverment (the below query IS NOT CORRECT !!!)
43 SELECT GovernmentForm
44 FROM Country ;
45
46 -- Hmmm, the above query keeps multiple copies of the same value.
47 -- Add the DISTINCT keyword to remove all the duplicates
48 SELECT DISTINCT GovernmentForm
49 FROM Country ;
50
51 -- We can use ORDER BY to order the city population by decreasing population
```

length : 19,678 lines : 691 Ln:51 Col:77 Sel:0|0

Windows (CR LF) UTF-8

INS

```
51 -- We can use ORDER BY to order the city population by decreasing population
52 SELECT Name, (Population / 1000000) AS PopulationInMillion
53 FROM City
54 WHERE Population >= '5000000'
55 ORDER BY PopulationInMillion DESC;
56
57 -- Find the top two most populated cities!
58 -- Here we can use LIMIT to limit the output
59 SELECT Name, (Population / 1000000) AS PopulationInMillion
60 FROM City
61 ORDER BY PopulationInMillion DESC
62 LIMIT 2;
63
64 -- Which is the first country that became independent?
65 -- The query below is NOT CORRECT! NULL values are present
66 SELECT Name, IndepYear
67 FROM Country
68 ORDER BY IndepYear ASC
69 LIMIT 1;
70
71 -- Instead the correct way is to filter out the NULL values by using NOT NULL
72 SELECT Name, IndepYear
73 FROM Country
74 WHERE IndepYear NOT NULL
75 ORDER BY IndepYear ASC
76 LIMIT 1;
77
78 -----
79 -- Multi-relational queries
80
81 -- What are the names of all countries that speak Greek?
82 -- Here we need to join two relations
83 SELECT Name
84 FROM Country, CountryLanguage
85 WHERE Code = CountryCode AND Language = 'Greek';
86
87 -- Better style for the above query
88 SELECT Country.Name
89 FROM Country, CountryLanguage
90 WHERE Country.Code = CountryLanguage.CountryCode
91 AND CountryLanguage.Language = 'Greek';
92
93 -- or even better (prefer this one!)
94 SELECT C.Name
95 FROM Country C, CountryLanguage L
96 WHERE C.Code = L.CountryCode
97 AND L.Language = 'Greek';
98
99 -- Which countries speaks at least 50% Greek?
100 SELECT C.Name
101 FROM Country C, CountryLanguage L
```

length: 19,678 lines: 691

Ln: 51 Col: 77 Sel: 0 | 0

Windows (CR LF) UTF-8

INS

Normal text file



```
100 SELECT C.Name
101 FROM Country C, CountryLanguage L
102 WHERE C.Code = L.CountryCode
103 AND L.Language = 'Greek'
104 AND L.Percentage >= 50 ;
105
106 -- What is the district of the capital of USA?
107 SELECT T.district
108 FROM Country C, City T
109 WHERE C.code = 'USA'
110 AND C.capital = T.id ;
111
112 -- Which countries speak Greek and English?
113 SELECT C.Name
114 FROM Country C, CountryLanguage L1, CountryLanguage L2
115 WHERE C.Code = L1.CountryCode
116 AND C.Code = L2.CountryCode
117 AND L1.Language = 'Greek'
118 AND L2.Language = 'English';
119
120 -- Alternatively we could also use the set operator INTERSECT
121 SELECT C.Name
122 FROM Country C, CountryLanguage L
123 WHERE C.Code = L.CountryCode
124 AND L.Language = 'Greek'
125 INTERSECT
126 SELECT C.Name
127 FROM Country C, CountryLanguage L
128 WHERE C.Code = L.CountryCode
129 AND L.Language = 'English' ;
130
131
132 -- Countries that have population more than 100 million and do not
133 -- speak English: here we use the set operator EXCEPT
134 SELECT C.Name
135 FROM Country C
136 WHERE C.Population >= 100000000
137 EXCEPT
138 SELECT C.Name
139 FROM Country C, CountryLanguage L
140 WHERE C.Code = L.CountryCode
141 AND L.Language = 'English' ;
142
143
144 -----
145 -- Nested queries
146
147 -- In which country is Berlin?
148 SELECT C.Name
149 FROM Country C
150 WHERE C.code =
```

length: 19,678 lines: 691

Ln: 51 Col: 77 Sel: 0 | 0

Windows (CR LF) UTF-8

INS



```
150 WHERE C.code =
151   (SELECT C.CountryCode
152    FROM City C
153    WHERE C.name = 'Berlin');
154
155 -- Find all countries in Europe with population more than 50 million
156 SELECT C.Name
157 FROM (SELECT Name, Continent
158          FROM Country
159         WHERE Population >50000000) AS C
160 WHERE C.Continent = 'Europe';
161
162 -- Find all countries in Europe that have *some* city with population
163 -- more than 5 million
164 SELECT C.Name
165 FROM Country C
166 WHERE C.Continent = 'Europe'
167 AND C.Code IN (SELECT CountryCode
168                  FROM City
169                 WHERE Population > 5000000);
170
171 -- alternatively:
172 SELECT C.Name
173 FROM Country C
174 WHERE C.Continent = 'Europe'
175 AND EXISTS (SELECT *
176               FROM City T
177              WHERE T.Population > 5000000
178             AND T.CountryCode = C.Code) ;
179
180 -- alternatively (the query below is NOT supported in SQLite!)
181 SELECT C.Name
182 FROM Country C
183 WHERE C.Continent = 'Europe'
184 AND 5000000 > ANY (SELECT T.Population
185                      FROM City T
186                     WHERE T.CountryCode = C.Code) ;
187
188 -- Find all countries in Europe that have all cities with population
189 -- less than 1 million
190 SELECT C.Name
191 FROM Country C
192 WHERE C.Continent = 'Europe'
193 AND NOT EXISTS (SELECT * FROM City T
194                      WHERE T.Population > 1000000
195                     AND T.CountryCode = C.Code) ;
196
197 -- alternatively (the query below is NOT supported by SQLite)
198 SELECT C.Name
199 FROM Country C
200 WHERE C.Continent = 'Europe'
```

length: 19,678 lines: 691 Ln:51 Col:77 Sel:0|0 Windows (CR LF) UTF-8 INS

```
197 -- alternatively (the query below is NOT supported by SQLite)
198 SELECT C.Name
199 FROM Country C
200 WHERE C.Continent = 'Europe'
201 AND 1000000 > ALL (SELECT T.Population
202     FROM City T
203     WHERE T.CountryCode = C.Code) ;
204
205
206
207 -----
208 -- Aggregate queries
209
210 -- Find the average population of countries in Europe
211 SELECT AVG(Population)
212 FROM Country
213 WHERE Continent = 'Europe';
214
215 -- Or max population:
216 SELECT MAX(Population)
217 FROM Country
218 WHERE Continent = 'Europe';
219
220 -- How many countries are in Europe?
221 SELECT COUNT(*)
222 FROM Country
223 WHERE Continent = 'Europe';
224
225 -- How many languages are spoken in the USA?
226 SELECT COUNT(*)
227 FROM CountryLanguage
228 WHERE CountryCode = 'USA';
229
230 -- Count the number of languages (this will NOT give the correct answer!!!!)
231 SELECT COUNT(Language)
232 FROM CountryLanguage ;
233
234 -- Instead use DISTINCT to eliminate duplicates:
235 SELECT COUNT(DISTINCT Language)
236 FROM CountryLanguage ;
237
238 -- Find the name and population of the country with the max population in Europe!
239 -- The example below is not correct! Why?
240 SELECT Name, MAX(Population)
241 FROM Country
242 WHERE Continent = 'Europe';
243
244 -- Instead we have two alternatives:
245 -- # 1: Use subqueries to do this!
246 SELECT Name, Population
247 FROM Country
```

length:19,678 lines:691 Ln:51 Col:77 Sel:0|0

Windows (CRLF) UTF-8

INS

```
244 -- Instead we have two alternatives:  
245 -- # 1: Use subqueries to do this!  
246 SELECT Name, Population  
247 FROM Country  
248 WHERE Population =  
249   (SELECT MAX(Population)  
250    FROM Country  
251    WHERE Continent = 'Europe');  
252  
253 -- # 1: Use order by and limit!  
254 SELECT Name, Population  
255 FROM Country  
256 WHERE Continent = 'Europe'  
257 ORDER BY Population DESC  
258 LIMIT 1 ;  
259  
260 -- Find how many countries have each form of government:  
261 SELECT GovernmentForm, COUNT(Code)  
262 FROM Country  
263 GROUP BY GovernmentForm ;  
264  
265 -- Find how many countries speak each language (with percentage > 50%) and  
266 -- output them in decreasing order!  
267 SELECT Language, COUNT(CountryCode) AS N  
268 FROM CountryLanguage  
269 WHERE Percentage >= 50  
270 GROUP BY Language  
271 ORDER BY N DESC ;  
272  
273 -- Find languages that are spoken in at least 3 different countries with  
274 -- percentage at least 50  
275 SELECT Language, COUNT(CountryCode) AS N  
276 FROM CountryLanguage  
277 WHERE Percentage >= 50  
278 GROUP BY Language  
279 HAVING N > 2  
280 ORDER BY N DESC ;  
281  
282 -- The query below is NOT semantically correct!  
283 -- HAVING clause is not correctly applied  
284 SELECT Language, COUNT(CountryCode) AS N  
285 FROM CountryLanguage  
286 GROUP BY Language  
287 HAVING Percentage > 50 ;  
288  
289 -- Output for each country the population most populated city,  
290 -- for countries with at least 10 cities!  
291 SELECT C.NAME AS Country, MAX(T.Population) AS N  
292 FROM City T, Country C  
293 WHERE C.Code = T.CountryCode  
294 GROUP BY C.Name
```

```
289 -- Output for each country the population most populated city,  
290 -- for countries with at least 10 cities!  
291 SELECT C.NAME AS Country, MAX(T.Population) AS N  
292 FROM City T, Country C  
293 WHERE C.Code = T.CountryCode  
294 GROUP BY C.Name  
295 HAVING COUNT(T.ID) > 9  
296 ORDER BY N DESC ;  
297  
298 -- How do we get the name of the most populated city as well?  
299 SELECT Temp.Country, T.Name  
300 FROM (SELECT C.NAME AS Country, C.Code AS Code, MAX(T.Population) AS N  
301     FROM City T, Country C  
302     WHERE C.Code = T.CountryCode  
303     GROUP BY C.Name, C.Code  
304     HAVING COUNT(T.ID) > 9) AS Temp, City T  
305 WHERE T.Population = Temp.N  
306 AND T.CountryCode = Temp.Code;  
307  
308 -----  
309 -- NULL behavior  
310  
311 SELECT COUNT(*)  
312 FROM Country  
313 WHERE IndepYear > 1990 OR IndepYear <= 1990 ;  
314  
315 -- Some countries are missing! What can we do?  
316 SELECT COUNT(*)  
317 FROM Country  
318 WHERE IndepYear > 1990 OR IndepYear <= 1990 OR IndepYear IS NULL;  
319  
320  
321 -- Returns max population of a city for each country, and for countries  
322 -- without any city returns NULL  
323 SELECT C.Name AS Country, MAX(T.Population) AS N  
324 FROM Country C LEFT OUTER JOIN City T ON C.Code = T.CountryCode  
325 GROUP BY C.Name  
326 ORDER BY N DESC ;  
327  
328 -----  
329 -- DB Modifications  
330  
331 -- Inserting a new tuple!  
332 INSERT INTO CountryLanguage VALUES('USA', 'C++', 'F', 0.5);  
333  
334 -- Let's delete it now.  
335 DELETE FROM CountryLanguage WHERE LAnguege = 'C++';  
336  
337 -- Let's update something  
338 UPDATE CountryLanguage  
339 SET IsOfficial = 'T'
```

length : 19,678 lines : 691 Ln: 51 Col: 77 Sel: 0 | 0

Windows (CR LF) UTF-8

INS

```
337 -- Let's update something
338 UPDATE CountryLanguage
339 SET IsOfficial = 'T'
340 WHERE CountryCode = 'USA' AND Language = 'Spanish';
341
342 -----
343 -- Views
344
345 -- Let's create a view that stores the name, and the official languages for each country
346 CREATE VIEW OfficialCountryLanguage AS
347 SELECT C.Name AS CountryName, L.Language AS Language
348 FROM CountryLanguage L, Country C
349 WHERE L.CountryCode = C.Code
350 AND L.IsOfficial = 'T' ;
351
352 -- Now we can use this view! Find the countries with more than one official language!
353 SELECT CountryName, COUNT(Language)
354 FROM OfficialCountryLanguage
355 GROUP BY CountryName
356 HAVING COUNT(Language) > 1 ;
357
358 ++++++
359
360 select date_sub(Now(), Interval 6 month); //select 6 month old date
361 //2018-03-13 15:09:12
362 SELECT * FROM test_users where user_created_date > date_sub(Now(), Interval 8 month);
363
364 Table: employee_details
365 Columns:
366 EmpId int(11) PK
367 FullName varchar(45)
368 ManagerId varchar(45)
369 DateOfJoining varchar(4)
370
371 Table: employee_salary
372 Columns:
373 EmpId int(11) PK
374 Project varchar(45)
375 Salary int(11)
376
377 -- get firstname of emp from fullname // ved prakash
378
379 SELECT SUBSTRING_INDEX(FullName, ' ', 1) FROM employee_details; //ved
380
381
382 -- Write a SQL query to fetch all the Employees who are also managers from EmployeeDetails table.
383
384 SELECT
385 A.EmpId,
386 A.FullName AS Emp,
387 B.ManagerId,
388 -----
Normal text file
```

```
382 -- Write a SQL query to fetch all the Employees who are also managers from EmployeeDetails table.
383
384 SELECT
385     A.EmpId,
386     A.FullName AS Emp,
387     B.ManagerId,
388     B.FullName AS Manager
389 FROM
390     employee_details AS A
391     JOIN
392     employee_details AS B ON B.EmpId = A.ManagerId;
393 -----
394 -- Write a SQL query to fetch all employee records from EmployeeDetails table who have a salary record in EmployeeSalary table.
395
396 SELECT * FROM employee_details E
397 WHERE EXISTS
398 (SELECT * FROM employee_salary S WHERE E.EmpId = S.EmpId);
399 -----
400 -- Write a SQL query to remove duplicates from a table without using temporary table.
401
402 DELETE FROM employee_salary
403 WHERE EmpId IN (
404     SELECT EmpId
405     FROM employee_salary
406     GROUP BY Project, Salary
407     HAVING COUNT(*) > 1);
408
409 -----
410 -- Write a SQL query to fetch only odd rows from table.
411
412 SET @row_number = 0;
413 SELECT
414     E.EmpId, E.Project, E.Salary
415 FROM
416     (SELECT *,
417         (@row_number:=@row_number + 1) AS RowNumber
418     FROM
419         employee_salary) E
420 WHERE
421     E.RowNumber % 2 = 1;
422 -----
423 -- Write a SQL query to fetch only even rows from table.
424
425 SET @row_number = 0;
426 SELECT
427     E.EmpId, E.Project, E.Salary
428 FROM
429     (SELECT *,
430         (@row_number:=@row_number + 1) AS RowNumber
431     FROM
432         employee_salary) E
```

length: 19,678 lines: 691 Ln: 51 Col: 77 Sel: 0 | 0 Windows (CR LF) UTF-8 INS



```
423 -- Write a SQL query to fetch only even rows from table.
424
425 SET @row_number = 0;
426 SELECT
427     E.EmpId, E.Project, E.Salary
428 FROM
429     (SELECT *,
430      (@row_number:=@row_number + 1) AS RowNumber
431     FROM
432         employee_salary) E
433 WHERE
434     E.RowNumber % 2 = 0;
435
436 -- Write a SQL query to create a new table with data and structure copied from another table.
437 create table Employee as select * from employee_details;
438
439
440 -- Write a SQL query to create an empty table with same structure as some other table.
441 create table Employee1 like employee_details;
442
443
444 mysql does not support intersect, minus
445
446 current datetime
447 select now()
448
449 -- Write a SQL query to fetch all the Employees from EmployeeDetails table who joined in Year 2016.
450
451 select * from employee_details where DateOfJoining between '01/01/2016' AND '31/12/2016';
452
453 -- Write a SQL query to fetch top n records?
454
455 select * from employee_details order by EmpId asc limit 6;
456
457 -- Write SQL query to find the nth highest salary from table. ex 5 th highest
458
459 select * from employee_salary order by Salary desc limit n-1, 1
460
461 -- Write SQL query to find the 5th highest salary from table without using TOP/limit keyword.
462
463 SELECT
464     *
465 FROM
466     employee_salary Emp1
467 WHERE
468     4 = (SELECT
469         COUNT(DISTINCT (Emp2.Salary))
470     FROM
471         employee_salary Emp2
472     WHERE
473         Emp2.Salary > Emp1.Salary)
```

```
461 -- Write SQL query to find the 5th highest salary from table without using TOP/limit keyword.
462
463 SELECT *
464   FROM
465     employee_salary Emp1
466 WHERE
467   4 = (SELECT
468       COUNT(DISTINCT (Emp2.Salary))
469     FROM
470       employee_salary Emp2
471     WHERE
472       Emp2.Salary > Emp1.Salary)
473
474
475
476 SELECT *
477   FROM Employee Emp1
478 WHERE (N-1) =
479   SELECT COUNT(DISTINCT(Emp2.Salary))
480     FROM Employee Emp2
481     WHERE Emp2.Salary > Emp1.Salary
482 -----
483 -- Select from table A which does not exist in table B
484
485 select A.*
486 from A left join B on A.BAND = B.HATE
487 where B.HATE IS NULL;
488 -----
489 create table and give reference of foreign key in other table
490
491 CREATE TABLE Worker (
492   WORKER_ID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
493   FIRST_NAME CHAR(25),
494   LAST_NAME CHAR(25),
495   SALARY INT(15),
496   JOINING_DATE DATETIME,
497   DEPARTMENT CHAR(25)
498 );
499
500 CREATE TABLE Bonus (
501   WORKER_REF_ID INT,
502   BONUS_AMOUNT INT(10),
503   BONUS_DATE DATETIME,
504   FOREIGN KEY (WORKER_REF_ID)
505     REFERENCES Worker(WORKER_ID)
506     ON DELETE CASCADE
507 );
508
509 ON DELETE CASCADE -> It specifies that the child data is deleted when the parent data is deleted.
510 ON UPDATE (Optional) -> It specifies what to do with the child data when the parent data is updated.
511
```

```
509 ON DELETE CASCADE -> It specifies that the child data is deleted when the parent data is deleted.
510 ON UPDATE (Optional) -> It specifies what to do with the child data when the parent data is updated.
511
512 You have the options of NO ACTION(no changes in child data),
513 CASCADE((child data is deleted or updated also),
514 SET NULL(child data is set to Null),
515 or SET DEFAULT(child data is set to their default values).
516
517 Worker TAble with data
518 WORKER_ID FIRST_NAME LAST_NAME SALARY JOINING_DATE DEPARTMENT
519 1 Monika Arora 100000 2/20/2014 9:00 HR
520 2 Niharika Verma 80000 6/11/2014 9:00 Admin
521 3 Vishal Singhal 300000 2/20/2014 9:00 HR
522 4 Amitabh Singh 500000 2/20/2014 9:00 Admin
523 5 Vivek Bhati 500000 6/11/2014 9:00 Admin
524 6 Vipul Diwan 200000 6/11/2014 9:00 Account
525 7 Satish Kumar 75000 1/20/2014 9:00 Account
526 8 Geetika Chauhan 90000 4/11/2014 9:00 Admin
527
528
529
530 //upper case
531 Select upper(FIRST_NAME) from Worker;
532
533 //first 3 character - substring
534 Select substring(FIRST_NAME,1,3) from Worker;
535
536 // check index of character: index of character a
537 Select INSTR(FIRST_NAME, BINARY'a') from Worker where FIRST_NAME = 'Amitabh'; // 5
538
539 //length of a field
540 Select distinct length(DEPARTMENT) from Worker;
541
542 //replace character
543 Select REPLACE(FIRST_NAME,'a','A') from Worker;
544
545 //concat
546 Select CONCAT(FIRST_NAME, ' ', LAST_NAME) AS 'COMPLETE_NAME' from Worker;
547
548 //end with h and contain 6 alphabets
549 Select * from Worker where FIRST_NAME like '_____h';
550
551 // year and month from date field , day, hour, minute,
552 Select * from Worker where year(JOINING_DATE) = 2014 and month(JOINING_DATE) = 2;
553
554 //show only odd rows from a table.
555 SELECT * FROM Worker WHERE MOD (WORKER_ID, 2) <> 0;
556
557 // even
558 SELECT * FROM Worker WHERE MOD (WORKER_ID, 2) = 0;
559
```

Normal text file

length : 19.678 lines : 691 Ln:51 Col:77 Sel:0|0 Windows (CR LF) UTF-8 INS



```
560 //clone a new table from another table
561 SELECT * INTO WorkerClone FROM Worker;
562 CREATE TABLE WorkerClone LIKE Worker;
563
564 //date
565 SELECT CURDATE(); //2019-05-21
566 SELECT NOW(); // 2019-05-21 17:59:36
567
568 //get employee with same salary
569 Select distinct W.WORKER_ID, W.FIRST_NAME, W.Salary
570 from Worker W, Worker W1
571 where W.Salary = W1.Salary
572 and W.WORKER_ID != W1.WORKER_ID;
573
574 //query to show one row twice in results from a table.
575 select FIRST_NAME, DEPARTMENT from worker W where W.DEPARTMENT='HR'
576 union all
577 select FIRST_NAME, DEPARTMENT from Worker W1 where W1.DEPARTMENT='HR';
578
579 // query to fetch the first 50% records from a table.
580 SELECT * FROM WORKER WHERE WORKER_ID <= (SELECT count(WORKER_ID)/2 from Worker);
581
582 //show the last record from a table.
583 Select * from Worker where WORKER_ID = (SELECT max(WORKER_ID) from Worker);
584
585 //select first row from table
586 Select * from Worker where WORKER_ID = (SELECT min(WORKER_ID) from Worker);
587
588 //SQL query to print the name of employees having the highest salary in each department.
589 SELECT
590     t.DEPARTMENT, t.FIRST_NAME, t.Salary
591 FROM
592     (SELECT
593         MAX(Salary) AS TotalSalary, DEPARTMENT
594     FROM
595         Worker
596     GROUP BY DEPARTMENT) AS TempNew
597     INNER JOIN
598         Worker t ON TempNew.DEPARTMENT = t.DEPARTMENT
599         AND TempNew.TotalSalary = t.Salary;
600
601
602 //get top 3 salary
603 SELECT DISTINCT
604     Salary
605 FROM
606     worker a
607 WHERE
608     3 >= (SELECT
609         COUNT(DISTINCT Salary)
610     FROM
```

length : 19,678 lines : 691 Ln:51 Col:77 Sel:0|0

Windows (CR LF) UTF-8

INS

```
602 //get top 3 salary
603 SELECT DISTINCT
604     Salary
605 FROM
606     worker a
607 WHERE
608     3 >= (SELECT
609         COUNT(DISTINCT Salary)
610     FROM
611         worker b
612     WHERE
613         a.Salary <= b.Salary)
614 ORDER BY a.Salary DESC;
615
616 // three min salaries from a table.
617 SELECT DISTINCT
618     Salary
619 FROM
620     worker a
621 WHERE
622     3 >= (SELECT
623         COUNT(DISTINCT Salary)
624     FROM
625         worker b
626     WHERE
627         a.Salary >= b.Salary)
628 ORDER BY a.Salary DESC;
629
630 //select name of workers with max salaries
631 SELECT FIRST_NAME, SALARY from Worker WHERE SALARY=(SELECT max(SALARY) from Worker);
632
633
634 Remove all duplicate row from table
635
636 with id in table
637 DELETE n1 FROM categories n1,
638     categories n2
639 WHERE
640     n1.id > n2.id AND n1.name = n2.name;
641
642 without id in table
643
644 -----
645 Difference between Stored Procedure and Function in SQL Server::
646
647 Both stored procedures and functions are database objects which contain a set of SQL statements to complete a task
648 Stored Procedures are pre-compiled objects which are compiled for the first time and its compiled format is saved, which executes (compiled code) whenever it is called
649
650 A function is compiled and executed every time whenever it is called.
651
652 -----
```

```
652 -----
653 Varchar vs varchar2 in oracle
654
655 VARCHAR is reserved by Oracle to support distinction between NULL and empty string .
656 VARCHAR2 does not distinguish between a NULL and empty string, and never will.
657 -----
658 Types of indexes: use to speed up data retrieval
659
660 Index:
661 CREATE INDEX index_name ON table_name;
662
663 Single column index:
664 CREATE INDEX index_name ON table_name (column_name);
665
666 Unique Indexes:
667 A unique index does not allow any duplicate values to be inserted into the table
668 CREATE UNIQUE INDEX index_name on table_name (column_name);
669
670 Composite Indexes:: on two or more columns of a table
671 CREATE INDEX index_name on table_name (column1, column2);
672
673 Drop index:
674 DROP INDEX index_name;
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
```

length: 19,734 lines: 719

Ln: 702 Col: 1 Sel: 010

Windows (CR LF) UTF-8

INS