--use UPPER and LOWER for display formatting... does not change table

SELECT UPPER(first\_name), LOWER(last\_name)

FROM employees;

--length retunrs numChars

SELECT LENGTH(first\_name), LOWER(last\_name)

FROM employees;

--use TRIM to get rid of leading, trailing spaces

SELECT TRIM(' HELLO WORLDS ');

--nested funtions work in to out

SELECT LENGTH(TRIM(' HELLO WORLDS '));

--use || to concatenate values from two columns together (and a space)

SELECT first\_name ||' '|| last\_name as full\_name --as keyword is cleaner but optional

FROM employees;

-- with another column

SELECT first\_name ||' '|| last\_name as full\_name, department

FROM employees;

--return booleans for certain returned display, false is default

SELECT first\_name ||' '|| last\_name as full\_name, (salary>115000)

FROM employees

ORDER BY salary>115000 DESC;

SELECT department, ('Clothing' IN (department))

FROM employees;

SELECT department, (department LIKE '%oth%')

FROM employees;

SELECT LENGTH(department), (department LIKE '%oth%')

FROM employees;

--extract substrings

SELECT 'This is test data' test\_data;

SELECT SUBSTRING('This is test data' FROM 1 FOR 7) sub\_string;

--use replace function to rename, etc strings

SELECT department, REPLACE(department, 'Clothing', 'Costumes') modified\_Data

FROM departments;

--new column and concat

SELECT department,

REPLACE(department, 'Clothing', 'Costumes') modified\_name,

department || ' department' AS "Complete Department Name"

FROM departments;

--use POSITION keyword to locate where a char occurs

SELECT POSITION('@' IN email)

FROM employees

WHERE salary>115000;

--Getting information from Strings

SELECT email, SUBSTRING(email, POSITION('@' IN email)+1) formatted\_text

FROM employees;

--coalesce fills a null cell

SELECT COALESCE(email, 'NO EMAIL') as email

FROM employees;

-- Grouping/Aggregate functions: not single row

SELECT MAX(salary)

FROM employees;

SELECT MIN(salary)

FROM employees;

SELECT AVG(salary)

FROM employees;

SELECT ROUND(AVG(salary))

FROM employees;

SELECt COUNT(employee\_id) -- primary key should return number of entries

FROM employees;

SELECT COUNT(email)

FROM employees;

SELECT SUM(salary)

FROM employees

WHERE department = 'Clothing';

-- Practice challenges with functions

SELECT last\_name ||' works in the '||department||' department'

FROM professors;

SELECT 'It is '||(salary>95000)||' that '||

last\_name||' is highly paid'

FROM professors;

SELECT \*, SUBSTRING(department FROM 1 FOR 3)

FROM professors;

SELECT last\_name, UPPER(SUBSTRING(department,1,3)) as department,

salary, hire\_date

FROM professors;

SELECT MIN(salary) as lowest\_salary, MAX(salary) as highest\_salary

from employees

WHERE last\_name != 'Wilson';

SELECT MIN(hire\_date)

FROM professors;

CREATE TABLE cars(make VARCHAR(10));

SELECT \* FROM cars;

INSERT INTO cars VALUES('HONDA');

INSERT INTO cars VALUES('HONDA');

INSERT INTO cars VALUES('TOYOTA');

INSERT INTO cars VALUES('HONDA');

INSERT INTO cars VALUES('TOYOTA');

INSERT INTO cars VALUES('NISSAN');

SELECT COUNT(\*)

FROM cars

GROUP BY(make);

INSERT INTO cars VALUES(Null);

INSERT INTO cars VALUES(Null);

INSERT INTO cars VALUES(Null);

INSERT INTO cars VALUES(Null);

SELECT make, COUNT(\*)

FROM cars

GROUP BY(make);

--GROUP BY in practice

SELECT department, AVG(salary)

FROM employees

WHERE salary > 100000

GROUP BY department;

SELECT department, COUNT(\*) number\_of\_employees, ROUND(AVG(salary)) avg\_sal, ROUND(MIN(salary)) min\_sal, ROUND(MAX(salary)) max\_sal

FROM employees

WHERE 1=1

GROUP BY department

ORDER BY number\_of\_employees desc;

SELECT department, COUNT(\*) number\_of\_employees, ROUND(AVG(salary)) avg\_sal, ROUND(MIN(salary)) min\_sal, ROUND(MAX(salary)) max\_sal

FROM employees

WHERE 1=1

GROUP BY department

ORDER BY avg\_sal desc;

SELECT department, gender, COUNT(gender) number

FROM employees

GROUP BY department, gender

ORDER BY department;

--cant do below because of salary not being aggregate function or part of group by

-- SELECT department, salary, COUNT(gender) number

-- FROM employees

-- GROUP BY department

-- ORDER BY department;

-- -- the below also doesnt work

-- SELECT department, COUNT(department) count

-- FROM employees

-- WHERE COUNT(department) > 35

-- GROUP BY department;

--use 'Having' for filter on agg functions

SELECT department, COUNT(department)

FROM employees

GROUP BY department

HAVING COUNT(department)>35

ORDER BY department;

--Practice Problems

--First names

SELECT first\_name, COUNT(first\_name) num\_name

FROM employees

GROUP BY first\_name

HAVING COUNT(first\_name) > 1

ORDER BY num\_name desc;

--unique departments without using 'distinct'

SELECT department

FROM employees

GROUP BY department;

--find domain name only for employees and count for each domain name

SELECT SUBSTRING(email, POSITION('@' IN email)+1) as domain, COUNT(SUBSTRING(email, POSITION('@' IN email)+1)) domain\_count

FROM employees

WHERE email IS NOT NULL--cannot use != for NULL

GROUP BY domain

ORDER BY domain\_count desc;

--can be accomplished using count(\*)

SELECT SUBSTRING(email, POSITION('@' IN email)+1) as domain, COUNT(\*) domain\_count

FROM employees

WHERE email IS NOT NULL--cannot use != for NULL

GROUP BY domain

ORDER BY domain\_count desc;

SELECT gender, region\_id, MIN(salary) min\_salary, MAX(salary) max\_salary, ROUND(AVG(salary)) avg\_salary

FROM employees

GROUP BY gender, region\_id

ORDER BY gender, region\_id;

--compare M and F for each region id

SELECT gender, region\_id, MIN(salary) min\_salary, MAX(salary) max\_salary, ROUND(AVG(salary)) avg\_salary

FROM employees

GROUP BY gender, region\_id

ORDER BY region\_id, gender;

--Assignment Problems

--script for schema

CREATE TABLE fruit\_imports

(

id integer,

name varchar(20),

season varchar(10),

state varchar(20),

supply integer,

cost\_per\_unit decimal

);

insert into fruit\_imports values(1, 'Apple', 'All Year', 'Kansas', 32900, 0.22);

insert into fruit\_imports values(2, 'Avocado', 'All Year', 'Nebraska', 27000, 0.15);

insert into fruit\_imports values(3, 'Coconut', 'All Year', 'California', 15200, 0.75);

insert into fruit\_imports values(4, 'Orange', 'Winter', 'California', 17000, 0.22);

insert into fruit\_imports values(5, 'Pear', 'Winter', 'Iowa', 37250, 0.17);

insert into fruit\_imports values(6, 'Lime', 'Spring', 'Indiana', 40400, 0.15);

insert into fruit\_imports values(7, 'Mango', 'Spring', 'Texas', 13650, 0.60);

insert into fruit\_imports values(8, 'Orange', 'Spring', 'Iowa', 18000, 0.26);

insert into fruit\_imports values(9, 'Apricot', 'Spring', 'Indiana', 55000, 0.20);

insert into fruit\_imports values(10, 'Cherry', 'Summer', 'Texas', 62150, 0.02);

insert into fruit\_imports values(11, 'Cantaloupe', 'Summer', 'Texas', 8000, 0.49);

insert into fruit\_imports values(12, 'Apricot', 'Summer', 'Kansas', 14500, 0.20);

insert into fruit\_imports values(13, 'Mango', 'Summer', 'Texas', 17000, 0.68);

insert into fruit\_imports values(14, 'Pear', 'Fall', 'Nebraska', 30500, 0.12);

insert into fruit\_imports values(15, 'Grape', 'Fall', 'Illinois', 72500, 0.35);

--view new table

SELECT \* FROM fruit\_imports;

--Only state with largest fruit supply

SELECT state, SUM(supply) sum

FROM fruit\_imports

GROUP BY state

ORDER BY sum desc

LIMIT 1;

--most expensive cost per unit of every season

SELECT season, MAX(cost\_per\_unit) max\_unit\_cost

FROM fruit\_imports

GROUP BY season

ORDER BY max\_unit\_cost desc;

--find state with more than one import of same fruit

SELECT state, COUNT(name)

FROM fruit\_imports

GROUP BY state, name

HAVING COUNT(name)>1;

--seasons with more than 2 fruits

SELECT season, COUNT(name)

FROM fruit\_imports

GROUP BY season

HAVING COUNT(name) >2;

-- state with total cost from supply and cost per unit

SELECT state, SUM(supply\*cost\_per\_unit) as total\_cost

FROM fruit\_imports

GROUP BY state

ORDER BY total\_cost desc

LIMIT 1;

--last problem

CREATE table fruits (fruit\_name varchar(10));

INSERT INTO fruits VALUES ('Orange');

INSERT INTO fruits VALUES ('Apple');

INSERT INTO fruits VALUES (NULL);

INSERT INTO fruits VALUES (NULL);

--display count of 4

SELECT COUNT(COALESCE(fruit\_name, 'fruit name'))

FROM fruits;