SQL - Basic SELECT statement

1.
   ```
   SELECT first_name AS ‘First Name’, last_name as ‘Last Name’
   FROM employees;
   ```

2.
   ```
   SELECT DISTINCT(department_id)
   FROM employees;
   ```

3.
   ```
   SELECT *
   FROM employees
   ORDER BY first_name DESC;
   ```

4.
   ```
   SELECT first_name, last_name, salary, 0.15*salary AS ‘PF’
   FROM employees;
   ```

5.
   ```
   SELECT employee_id, first_name, last_name, salary
   FROM employees
   ORDER BY salary ASC;
   ```

6.
   ```
   SELECT SUM(salary)
   FROM employees;
   ```

7.
   ```
   SELECT MAX(salary), MIN(salary)
   FROM employees;
   ```

8.
   ```
   SELECT AVG(salary), COUNT(DISTINCT(employee_id))
   FROM employees;
   ```

9.
   ```
   SELECT COUNT(*)
   FROM employees;
   ```

10.
    ```
    SELECT COUNT(DISTINCT(job_id))
    FROM employees;
    ```

11.
    ```
    SELECT UPPER(first_name)
    FROM employees;
    ```

12.
    ```
    SELECT SUBSTRING(first_name, 1, 3)
    FROM employees;
    ```

13.
    ```
    SELECT 171*214+625 AS result;
    ```

14.
    ```
    SELECT CONCAT(first_name, ‘ ‘, last_name) AS ‘EMPLOYEE NAME’
    FROM employees;
    ```

15.
    ```
    SELECT TRIM(first_name)
    FROM employees;
    ```

16.
    ```
    SELECT first_name, last_name, LENGTH(first_name) + LENGTH(last_name) as
    ‘LENGTH OF NAMES’
    FROM employees;
    ```

17.
    ```
    SELECT *
    FROM employees
    WHERE first_name REGEXP '[0-9]';
    ```

18.
    ```
    SELECT employee_id, first_name
    FROM employees
    LIMIT 10;
    ```

19.
    ```
    SELECT first_name, last_name, ROUND(salary/12, 2) as ‘Monthly Salary’
    FROM employees;
    ```

<u>SQL - Boolean & Relational Operators</u>

1.
   ```
   SELECT *
   FROM customer
   WHERE grade > 100
   ```

2.
   ```
   SELECT *
   FROM customer
   WHERE city = ‘New York’
   AND grade > 100
   ```

3.
   ```
   SELECT *
   FROM customer
   WHERE city = ‘New York’
   OR grade > 100
   ```

4.
   ```
   SELECT *
   FROM customer
   ```

```
   WHERE city = ‘New York’
   OR NOT grade > 100
```

5.
```
SELECT *
FROM customer
WHERE NOT (city = ‘New York’
OR grade > 100)
```

6.
```
SELECT *
FROM orders
WHERE NOT (ord_date = ‘2012-09-10’
AND salesman_id > 5005
OR purch_amt > 1000)
```

7.
```
SELECT *
FROM salesman
WHERE commission BETWEEN 0.09 AND 0.13
```

8.
```
SELECT *
FROM orders
WHERE purch_amt < 200
OR NOT (ord_date >= ‘2012-02-10’
AND customer_id < 3009)
```

9.
```
SELECT *
FROM orders
WHERE NOT ((ord_date = ‘2012-08-17’
OR customer_id > 3005)
AND purch_amt < 1000)
```

10.
```
SELECT ord_no, purch_amt,
(100*purch_amt)/6000 AS "Achieved %",
(100*(6000-purch_amt)/6000) AS "Unachieved %"
FROM orders
WHERE (100*purch_amt)/6000>50
```

11.
```
SELECT *
FROM emp_details
WHERE EMP_LNAME = ‘Dosni’
OR EMP_LNAME = ‘Mardy’
```

12. SELECT *
    FROM emp_details
    WHERE EMP_DEPT = 47
    OR EMP_DEPT = 63

<u>SQL - Wildcard and Special operators</u>

1. SELECT *
   FROM salesman
   WHERE city = ‘Paris’
   OR city = ‘Rome’

2. SELECT *
   FROM salesman
   WHERE city IN (‘Paris’, ‘Rome’)

3. SELECT *
   FROM salesman
   WHERE city NOT IN (‘Paris’, ‘Rome’)

4. SELECT *
   FROM customer
   WHERE customer_id BETWEEN 3006 AND 3010

5. SELECT *
   FROM salesman
   WHERE commission BETWEEN 0.11 AND 0.15

6. SELECT *
   FROM orders
   WHERE purch_amt BETWEEN 499 AND 4001
   AND purch_amt NOT IN (948.50, 1983.43)

7. SELECT *
   FROM SALESMAN
   WHERE name BETWEEN ‘A’ AND ‘L’

8.
```
SELECT *
FROM SALESMAN
WHERE name NOT BETWEEN ‘A’ AND ‘L’
```

9.
```
SELECT *
FROM customer
WHERE cust_name LIKE ‘B%’
```

10.
```
SELECT *
FROM customer
WHERE cust_name LIKE ‘%n’
```

11.
```
SELECT *
FROM salesman
WHERE name LIKE ‘N__l%’
```

12.
```
SELECT *
FROM testtable
WHERE col1 LIKE '%/_%' ESCAPE '/'
```

13.
```
SELECT *
FROM testtable
WHERE col1 NOT LIKE '%/_%' ESCAPE '/'
```

14.
```
SELECT *
FROM testtable
WHERE col1 LIKE ‘%//%’ ESCAPE ‘/’
```

15.
```
SELECT *
FROM testtable
WHERE col1 NOT LIKE ‘%//%’ ESCAPE ‘/’
```

16.
```
SELECT *
FROM testtable
WHERE col1 LIKE ‘%_//%’ ESCAPE ‘/‘
```

17.
```
SELECT *
FROM testtable
WHERE col1 NOT LIKE ‘%_//%’ ESCAPE ‘/‘
```

18. SELECT *
    FROM testtable
    WHERE col1 LIKE '%/%%' ESCAPE '/'

19. SELECT *
    FROM testtable
    WHERE col1 NOT LIKE '%/%%' ESCAPE '/'

20. SELECT *
    FROM customer
    WHERE grade IS NULL

21. SELECT *
    FROM customer
    WHERE grade IS NOT NULL

22. SELECT *
    FROM emp_details
    WHERE emp_lname LIKE 'D%'

SQL JOINS (HR DATABASE)

1. SELECT E.first_name, E.last_name, D.department_id, D.department_name
   FROM departments D
   JOIN employees E
   ON D.department_id = E.department_id;

2. SELECT E.first_name,E.last_name, D.department_name, L.city, L.state_province
   FROM employees E
   JOIN departments D
   ON E.department_id = D.department_id
   JOIN locations L
   ON D.location_id = L.location_id;

3. SELECT E.first_name, E.last_name, E.salary, J.grade_level
   FROM employees E
   JOIN job_grades J
   ON E.salary BETWEEN J.lowest_sal AND J.highest_sal;

4.
   ```
   SELECT E.first_name, E.last_name, D.department_id, D.department_name
   FROM departments D
   JOIN employees E
   ON D.department_id = E.department_id
   AND D.department_id IN (40, 80)
   ORDER BY E.last_name ASC;
   ```

5.
   ```
   SELECT E.first_name, E.last_name. D.department_name, L.city,
   L.state_province
   FROM departments D
   JOIN employees E
   ON D.department_id = E.department_id
   JOIN locations L
   ON D.location_id = L.location_id
   WHERE first_name LIKE ‘%z%’
   ```

6.
   ```
   SELECT E.first_name, E.last_name, D.department_id, D.department_name
   FROM employees E
   RIGHT OUTER JOIN department
   ON E.department_id = D.department_id
   ```

7.
   ```
   SELECT E.first_name, E.last_name, E.salary
   FROM employees E
   JOIN employee S
   ON E.salary < S.salary
   WHERE S.employee_id = 182
   ```

8.
   ```
   SELECT E.first_name, M.first_name
   FROM employees E
   JOIN employees M
   ON E.manager_id = M.employee_id
   ```

9.
   ```
   SELECT D.department_name , L.city , L.state_province
   FROM  departments D
   JOIN locations L
   ON  D.location_id = L.location_id;
   ```

10. SELECT E.first_name, E.last_name, D.department_id, D.department_name
    FROM employees E
    LEFT OUTER JOIN department D
    ON E.department_id = D.department_id

11. SELECT E.first_name AS ‘Employee name’,
    M.first_name AS ‘Manager’
    FROM employees E
    LEFT OUTER JOIN employees M
    ON E.manager_id = M.employee_id

12. SELECT E.first_name, E.last_name, E.department_id
    FROM employees E
    JOIN employees T
    ON E.department_id = T.department_id
    AND T.last_name = ‘Taylor’

13. SELECT J.job_title, D.department_name, CONCAT(first_name,’ ’ ,last_name) AS full_name, H.hire_date
    FROM job_history H
    JOIN employees E
    ON H.employee_id = E.employee_id
    JOIN jobs J
    ON E.job_id = J.job_id
    JOIN departments D
    ON E.department_id = D.department_id
    WHERE start_date BETWEEN ‘1993-01-01’ AND ‘1997-08-31’

14. SELECT job_title, CONCAT(first_name,’ ’ ,last_name) AS full_name, max_salary - salary AS salary_difference
    FROM employees
    NATURAL JOIN jobs

15. SELECT department_name, AVG(salary) AS average_salary,
    COUNT(DISTINCT(employee_id))
    FROM employees
    NATURAL JOIN departments
    GROUP BY department_name

16. SELECT job_title, CONCAT(first_name,' ' ,last_name) AS full_name, max_salary - salary AS salary_difference
    FROM employees
    NATURAL JOIN jobs
    WHERE department_id = 80

17. SELECT country_name, city, department_name
    FROM countries
    NATURAL JOIN locations
    NATURAL JOIN departments

18. SELECT department_name, CONCAT(first_name,' ' ,last_name) AS full_name
    FROM departments D
    JOIN employees E
    ON D.manager_id = E.manager_id

19. SELECT job_title, AVG(salary)
    FROM employees
    NATURAL JOIN jobs
    GROUP BY job_title

20. SELECT employee_id, start_date, end_date, job_id, department_id
    FROM employees E
    JOIN job_history J
    ON E.employee_id = J.employee.id
    WHERE salary > 12000

21. SELECT country_name,city, COUNT(department_id)
    FROM countries
    NATURAL JOIN locations
    NATURAL JOIN departments
    WHERE department_id IN
    (SELECT department_id
    FROM employees
    GROUP BY department_id
    HAVING COUNT(department_id)>=2)
    GROUP BY country_name,city;

22. SELECT department_name, CONCAT(first_name,’ ’ ,last_name) AS full_name, city
FROM employees E
JOIN departments D
ON E.employee_id = D.manager_id
JOIN locations L
ON D.location_id = L.location_id

23. SELECT employee_id, job_title, DATEDIFF(day, start_date, end_date) AS days
FROM jobs
NATURAL JOIN job_history
WHERE department_id = 80

24. SELECT CONCAT(first_name,’ ’ ,last_name) AS employee_name, salary
FROM employees E
JOIN departments D
ON E.department_id = D.department_id
JOIN locations L
ON D.location_id = L.location_id
WHERE city = ‘London’

25. SELECT CONCAT(e.first_name, ' ', e.last_name) AS Employee_name, j.job_title, h.*
FROM employees e
JOIN
(SELECT MAX(start_date),
MAX(end_date),
employee_id
FROM job_history
GROUP BY employee_id) h ON e.employee_id=h.employee_id
JOIN jobs j ON j.job_id=e.job_id
WHERE e.commission_pct = 0

26. SELECT department_name, department_id, COUNT(employee_id) as no_of_employees
FROM departments D
JOIN employees E
ON D.department_id = E.department_id
GROUP BY department_id

27. SELECT CONCAT(e.first_name, ' ', e.last_name) AS employee_name, c.country_id, c.country_name
FROM employees e
JOIN departments d using (department_id)
JOIN locations l using (location_id)
JOIN countries c using (country_id)

<u>SQL - SORTING and FILTERING on HR Database</u>

1. SELECT CONCAT(first_name, ‘ ’, last_name) as full_name, salary
FROM employees
WHERE salary < 6000

2. SELECT first_name, last_name, department_id, salary
FROM employees
WHERE salary > 8000

3. SELECT first_name, last_name, department_id
FROM employees
WHERE last_name = ‘McEwen’

4. SELECT *
FROM employees
WHERE department_id IS NULL

5. SELECT *
FROM department
WHERE department_name = ‘Marketing’

6. SELECT CONCAT(first_name, ‘ ’, last_name) as full_name, hire_date, salary, department_id
FROM employees
WHERE first_name NOT LIKE ‘%M%’
ORDER BY department_id ASC

7.
```
SELECT *
FROM employees
(WHERE salary IN (8000, 12000)
AND commission_pct > 0.00)
OR
(WHERE hire_date < ‘2003-06-05’
AND department_id NOT IN (40, 120,70))
```

8.
```
SELECT CONCAT(first_name, ‘ ’, last_name) as full_name, salary
FROM employees
WHERE commission_pct IS NULL
```

9.
```
SELECT CONCAT(first_name, ‘ ’, last_name) as full_name, CONCAT(phone_
number, ‘ ’, ‘-’, ‘ ’, email)
FROM employees
WHERE salary BETWEEN 8999 AND 17001
```

10.
```
SELECT first_name, last_name, salary
FROM employees
WHERE first_name LIKE ‘%m’
```

11.
```
SELECT CONCAT(first_name, ‘ ’, last_name) as name, salary
FROM employees
WHERE salary NOT BETWEEN 6999 AND 15001
ORDER BY CONCAT(first_name, ‘ ’, last_name)
```

12.
```
SELECT CONCAT(first_name, ‘ ’, last_name) as full_name, job_id, hire_date
FROM employees
WHERE hire_date
BETWEEN ‘2007-11-04’ AND ‘2009-07-06’
```

13.
```
SELECT CONCAT(first_name, ‘ ’, last_name) as full_name, department_id
WHERE department_id IN (70, 90)
```

14.
```
SELECT CONCAT(first_name, ‘ ’, last_name) as full_name, salary, manager_id
FROM employees
WHERE manager_id IS NOT NULL
```

15.
```
SELECT *
FROM employees
```

```
    WHERE hire_date < ‘2002-06-21’

16. SELECT first_name, last_name, email, salary, manager_id
    FROM employees
    WHERE manager_id IN (120, 103, 145)

17. SELECT *
    FROM employees
    WHERE first_name LIKE ‘%D%’
    OR first_name LIKE ‘%S%’
    OR first_name LIKE ‘%N%’
    ORDER BY salary DESC

18. SELECT first_name ||' '|| last_name AS full_name, hire_date, commission_pct,
    email ||'-'|| phone_number AS contact_details, salary
    FROM employees
    WHERE salary > 11000
    OR phone_number LIKE '______3%'
    ORDER BY first_name DESC

19. SELECT first_name, last_name, department_id
    FROM employees
    WHERE first_name LIKE ‘__s%’

20. SELECT employee_id, first_name, job_id, department_id
    FROM employees
    WHERE department_id NOT IN (50, 30, 80)

21. SELECT employee_id, first_name, job_id, department_id
    FROM employees
    WHERE department_id IN (90, 30, 40)

22. SELECT employee_id
    FROM job_history
    GROUP BY employee_id
    HAVING COUNT(employee_id) > 1
```

23. SELECT job_id, COUNT(employee_id), SUM(salary), MAX(salary) - MIN(salary)
AS salary_difference
FROM employees
GROUP BY job_id

24. SELECT job_id
FROM job_history
WHERE COUNT(DATEDIFF(day, end_date, start_date) > 300) > 2
GROUP BY job_id

25. SELECT country_id, COUNT(city)
FROM locations
GROUP BY country_id

26. SELECT manager_id, COUNT(*)
FROM employees
GROUP BY manager_id

27. SELECT *
FROM jobs
ORDER BY job_title DESC

28. SELECT first_name, last_name, hire_date
FROM employees
WHERE job_id IN (‘ST_MAN’, ‘SA_REP’)

29. SELECT department_id, AVG(salary)
FROM employees
WHERE commission_pct IS NOT NULL
GROUP BY department_id

30. SELECT DISTINCT department_id
FROM employees
GROUP BY department_id, manager_id
HAVING COUNT(employee_id) >= 4

31. SELECT DISTINCT department_id
    FROM employees
    GROUP BY department_id
    HAVING COUNT(employee_id) > 10
    AND commission_pct IS NOT NULL

32. SELECT employee_id , MAX(end_date)
    FROM job_history
    WHERE employee_id IN (SELECT employee_id
    FROM job_history
    GROUP BY 1
    HAVING COUNT(employee_id) > 1)
    GROUP BY 1

33. SELECT *
    FROM employees
    WHERE commission_pct IS NULL
    AND salary BETWEEN 7000 AND 12000
    AND department_id = 50

34. SELECT job_id, AVG(salary)
    FROM employees
    GROUP BY job_id
    HAVING AVG(salary) > 8000

35. SELECT job_title, max_salary - min_salary AS salary_difference
    FROM employees
    WHERE max_salary BETWEEN 12000 AND 18000

36. SELECT first_name, last_name
    FROM employees
    WHERE first_name LIKE ‘D%’
    OR last_name LIKE ‘D%’

37. SELECT *
    FROM jobs
    WHERE min_salary > 9000

38. SELECT *
    FROM employees
    WHERE hire_date > ‘1987-09-07’

SQL - MySQL Create Table

1. CREATE TABLE countries
   (country_id VARCHAR(2),
   country_name VARCHAR(40),
   region_id DECIMAL(10, 0))

2. CREATE TABLE IF NOT EXISTS countries
   (country_id VARCHAR(2),
   country_name VARCHAR(40),
   region_id DECIMAL(10, 0))

3. CREATE TABLE IF NOT EXISTS dup_countries
   LIKE countries

4. CREATE TABLE IF NOT EXISTS dup_countries
   AS SELECT * FROM countries

5. CREATE TABLE countries
   (country_id VARCHAR(2) NOT NULL,
   country_name VARCHAR(40) NOT NULL,
   region_id DECIMAL(10, 0)) NOT NULL)

6. CREATE TABLE jobs
   (job_id VARCHAR(10) NOT NULL,
   job_title VARCHAR(40) NOT NULL,
   min_salary DECIMAL(6, 0),
   max_salary DECIMAL(6, 0)
   CHECK(max_salary <= 25000)

7. CREATE TABLE countries
   (country_id VARCHAR(2) NOT NULL,
   country_name VARCHAR(40) NOT NULL,
   CHECK (country_name IN (‘Italy’, ‘India’, ‘China’))
   region_id DECIMAL(10, 0)) NOT NULL)

```
8.  CREATE TABLE job_histry
    (employee_id VARCHAR(10) NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL
    CHECK (end_date LIKE ‘--/--/----’),
    job_id VARCHAR(5) NOT NULL,
    department_id DECIMAL(5, 0) NOT NULL)

9.  CREATE TABLE countries
    (country_id VARCHAR(2) NOT NULL,
    country_name VARCHAR(40) NOT NULL,
    region_id DECIMAL(10, 0)) NOT NULL)
    UNIQUE(country_id)

10. CREATE TABLE jobs
    (job_id VARCHAR(10) NOT NULL UNIQUE,
    job_title VARCHAR(40) NOT NULL DEFAULT,
    min_salary DECIMAL(6, 0) DEFAULT 8000,
    max_salary DECIMAL(6, 0) DEFAULT NULL)

11. CREATE TABLE countries
    (country_id VARCHAR(2) NOT NULL UNIQUE PRIMARY_KEY,
    country_name VARCHAR(40) NOT NULL,
    region_id DECIMAL(10, 0)) NOT NULL)

12. CREATE TABLE countries
    (country_id VARCHAR(2) NOT NULL UNIQUE AUTO_INCREMENT PRIMARY
    KEY,
    country_name VARCHAR(40) NOT NULL,
    region_id DECIMAL(10, 0)) NOT NULL)

13. CREATE TABLE countries
    (country_id VARCHAR(2) NOT NULL UNIQUE DEFAULT ‘ ’,
    country_name VARCHAR(40) DEFAULT NULL,
    region_id DECIMAL(10, 0)) NOT NULL)
    PRIMARY KEY (country_id, region_id)
```

```
14. CREATE TABLE job_histry
    (employee_id VARCHAR(10) NOT NULL UNIQUE,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    job_id VARCHAR(5) NOT NULL,
    department_id DECIMAL(5, 0) DEFAULT NULL,
    FOREIGN KEY(job_id) REFERENCES jobs(job_id))
```

```
15. CREATE TABLE employees
    (employee_id VARCHAR(10) NOT NULL PRIMARY KEY,
    first_name VARCHAR(20) DEFAULT NULL,
    last_name VARCHAR(20) NOT NULL,
    email VARCHAR(50) NOT NULL UNIQUE,
    phone_number VARCHAR(15) DEFAULT NULL UNIQUE,
    hire_date DATE NOT NULL,
    job_id VARCHAR(5) NOT NULL,
    salary DECIMAL(8, 2) NOT NULL,
    commission DECIMAL(2, 2) NOT NULL,
    manager_id DECIMAL(6, 0) NOT NULL,
    department_id DECIMAL(4, 0) NOT NULL,
    FOREIGN KEY(department_id, manager_id) REFERENCES
    departments(department_id, manager_id))
```

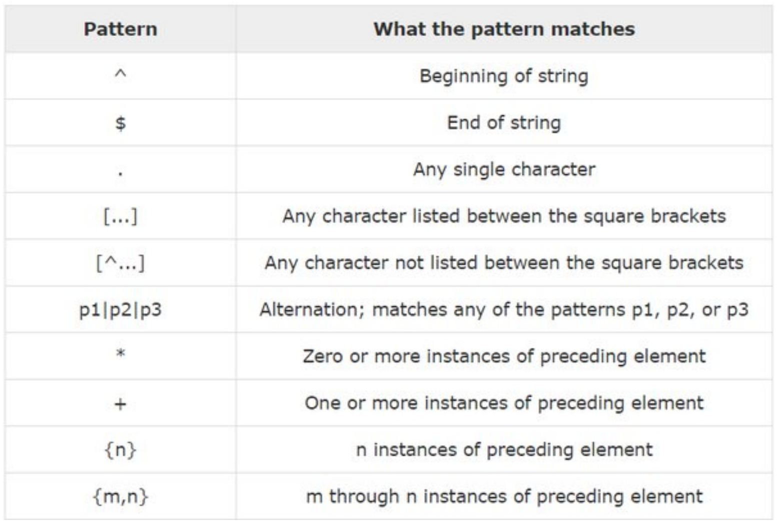
```
16. CREATE TABLE employees
    (employee_id VARCHAR(10) NOT NULL PRIMARY KEY,
    first_name VARCHAR(20) DEFAULT NULL,
    last_name VARCHAR(20) NOT NULL,
    email VARCHAR(50) NOT NULL UNIQUE,
    phone_number VARCHAR(15) DEFAULT NULL UNIQUE,
    hire_date DATE NOT NULL,
    job_id VARCHAR(5) NOT NULL,
    salary DECIMAL(8, 2) NOT NULL,
    commission DECIMAL(2, 2) NOT NULL,
    manager_id DECIMAL(6, 0) NOT NULL,
    department_id DECIMAL(4, 0) NOT NULL,
    FOREIGN KEY(department_id) REFERENCES departments(department_id),
    FOREIGN KEY(job_id) REFERENCES jobs(job_id)
```

New Findings

1. IFNULL(Column, Alternate_Value)

2. GROUP_CONCAT(Value) :Concatenate up a bunch of values in one column

3. REGEXP



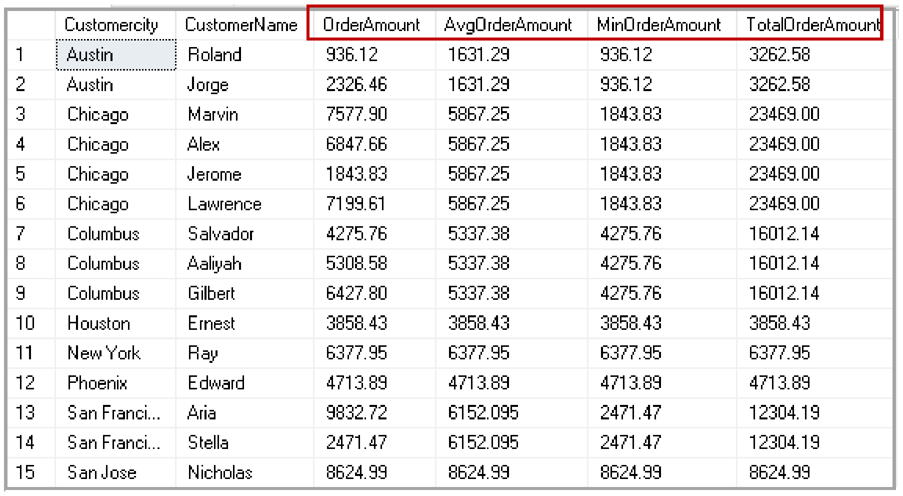
Pattern	What the pattern matches
^	Beginning of string
$	End of string
.	Any single character
[...]	Any character listed between the square brackets
[^...]	Any character not listed between the square brackets
p1|p2|p3	Alternation; matches any of the patterns p1, p2, or p3
*	Zero or more instances of preceding element
+	One or more instances of preceding element
{n}	n instances of preceding element
{m,n}	m through n instances of preceding element

4. SUBSTR(string, initial(num), final(num)) = extracting a few characters

5. PARTITION BY = GROUP BY (Number of rows isn't affected)

-Query Example:

```
SELECT Customercity,
AVG(Orderamount) OVER(PARTITION BY Customercity) AS AvgOrderAmount,
MIN(OrderAmount) OVER(PARTITION BY Customercity) AS MinOrderAmount,
SUM(Orderamount) OVER(PARTITION BY Customercity) TotalOrderAmount
FROM [dbo].[Orders];
```



	Customercity	CustomerName	OrderAmount	AvgOrderAmount	MinOrderAmount	TotalOrderAmount
1	Austin	Roland	936.12	1631.29	936.12	3262.58
2	Austin	Jorge	2326.46	1631.29	936.12	3262.58
3	Chicago	Marvin	7577.90	5867.25	1843.83	23469.00
4	Chicago	Alex	6847.66	5867.25	1843.83	23469.00
5	Chicago	Jerome	1843.83	5867.25	1843.83	23469.00
6	Chicago	Lawrence	7199.61	5867.25	1843.83	23469.00
7	Columbus	Salvador	4275.76	5337.38	4275.76	16012.14
8	Columbus	Aaliyah	5308.58	5337.38	4275.76	16012.14
9	Columbus	Gilbert	6427.80	5337.38	4275.76	16012.14
10	Houston	Ernest	3858.43	3858.43	3858.43	3858.43
11	New York	Ray	6377.95	6377.95	6377.95	6377.95
12	Phoenix	Edward	4713.89	4713.89	4713.89	4713.89
13	San Franci...	Aria	9832.72	6152.095	2471.47	12304.19
14	San Franci...	Stella	2471.47	6152.095	2471.47	12304.19
15	San Jose	Nicholas	8624.99	8624.99	8624.99	8624.99

6. Common Table Expression (CTE)

-Acts like a temporary table (Without constraints)

-Guide: WITH cte_name AS (Query *Data needed from a certain table*)

(Query *Data needed from the CTE*)

-Example:

```
WITH cte_name AS (
SELECT column_name FROM table
WHERE condition
)
SELECT * FROM cte_name
```

7.NATURAL JOIN = naturally join two tables based on what the computer found similar.

8. Temporary Table = CTE but with constraints

-Query Example:

```
CREATE TEMPORARY TABLE table_name(
   column_1_definition,
   column_2_definition,
   ...,
   table_constraints
);
INSERT INTO TEMPORARY TABLE (
VALUE, VALUE
)
```

-Taking data from other tables into temp table:

```
INSERT INTO temporary_table
SELECT *
FROM table
```

9. TRIM: TRIM, LTRIM, RTRIM

-Cuts off the blank space

-Example:

“ 69420” > TRIM(“ 69420”) > “69420”

10. REPLACE

SYNTAX: REPLACE(column_name, ‘origin value’, ‘updated value’)

11. SUBSTRING

SYNTAX: SUBSTRING(column_name, first letter index, extract the next amount character)

EXAMPLE: SUBSTRING(“hello”, 2, 3) > “ell”

12. UPPER, LOWER

-Upper case and lower case

13. STORED PROCEDURE

-To store a query, to use over and over again

-SYNTAX:

```
CREATE PROCEDURE procedure_name
AS
Query…….
```

-To run a stored procedure:

```
RUN procedure_name
```

-To edit what’s inside the stored procedure:

```
ALTER procedure_name
```

14. INDEX

-Make a faster data search on a specific column

-SYNTAX:

```
CREATE INDEX index_name
ON table(column)
```

15. BACKUP

-Backup a database

-General SYNTAX:

```
BACKUP DATABASE databasename
TO DISK = 'filepath';
```

-Differential Backup - Backup the only part that changes in a database

-Differential SYNTAX:

```
BACKUP DATABASE databasename
TO DISK = 'filepath'
WITH DIFFERENTIAL;
```

16. VIEW

-Privacy of data, allowing others to see a data that can be shared to the public

-SYNTAX:

CREATE VIEW view_name AS

QUERY;

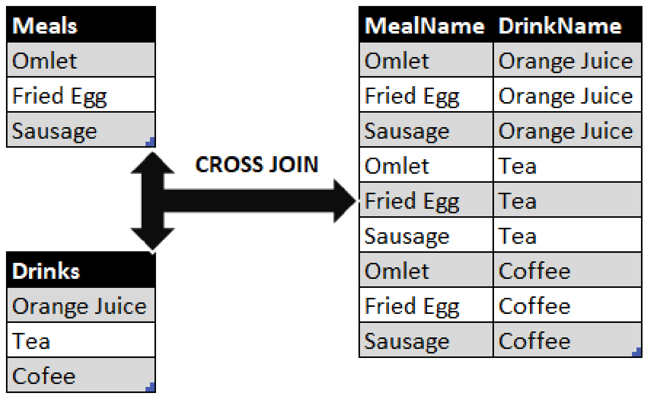
17. COALESCE

-Returns the first non-null value of a column

-SYNTAX:

SELECT COALESCE(QUERY)

18. CROSS JOIN



Meals
Omlet
Fried Egg
Sausage

Drinks
Orange Juice
Tea
Cofee

MealName	DrinkName
Omlet	Orange Juice
Fried Egg	Orange Juice
Sausage	Orange Juice
Omlet	Tea
Fried Egg	Tea
Sausage	Tea
Omlet	Coffee
Fried Egg	Coffee
Sausage	Coffee