* SELECT statement with WHERE filter and limited results with offset

SELECT Continent, Country, Population

FROM population\_data

WHERE Population > 100000

LIMIT 5

OFFSET 1

* Order the results

SELECT Continent, Country, Population

FROM population\_data

WHERE Population > 100000

ORDER BY Population DESC

* Rename a column (alias)

SELECT Continent, Country AS Region, Population

FROM population\_data

WHERE Population > 100000

* Multiple conditions with WHERE

SELECT Continent, Country, Population

FROM population\_data

WHERE

Continent = “Asia”

OR

Continent = “Europe”

SELECT Continent, Country, Population

FROM population\_data

WHERE

Continent = “Europe”

AND

Population > 100000

* Data aggregation with SUM, COUNT, MAX, MIN, AVERAGE

SELECT Continent, SUM(Population) AS Total\_population

FROM population\_data

GROUP BY Continent

SELECT Continent, COUNT(Country) AS Countries\_in\_continent

FROM population\_data

GROUP BY Continent

SELECT Continent, Country, MAX(Population)

FROM population\_data

GROUP BY Continent

HAVING Population > 50000

* Insert data into a table

INSERT INTO population\_data

(Continent, Country, Population)

VALUES (“Asia”, “India”, “1000000000”)

* Update table and ALTER table

Update Customers

SET FirstName = “Mangaljit”, LastName = “Singh”

WHERE id = 10

ALTER Customers

RENAME COLUMN “FirstName” TO “FName”

ALTER Customers

ADD COLUMN “Address” TEXT

* DELETE FROM

DELETE FROM Customers

WHERE id = 10

* Create a table with constraints (PRIMARY KEY, NOT NULL, UNIQUE, DEFAULT value)

CREATE TABLE Customers(

Id INTEGER PRIMARY KEY

FirstName TEXT,

LastName TEXT NOT NULL,

Age INTEGER.

)

* Drop a table

DROP TABLE IF EXISTS Customers

* Insert values from other table

INSERT INTO table1

VALUES

SELECT \* FROM table2

WHERE salary > 50000

* Access null values

SELECT \* FROM customers

WHERE

FirstName IS NULL

OR

LastName IS NULL

* Access not null values

SELECT \* FROM customers

WHERE

FirstName NOT NULL

* USE of LIKE

SELECT \* FROM population\_data

WHERE

Country LIKE “%stan”

SELECT \* FROM population\_data

WHERE

Country LIKE “A%”

* Use of IN

SELECT \* FROM population\_data

WHERE

Continent IN (“Asia”, “Europe”, “Africa”)

* CASE WHEN

SELECT FName, LName, Department, Salary,

CASE

WHEN Department = ‘IT’ THEN Salary = Salary + 0.3\*Salary,

WHEN Department = ‘Administration’ THEN Salary = Salary + 0.2\*Salary,

WHEN Department = ‘Design’ THEN Salary = Salary + 0.1\*Salary

END AS Salary\_after\_raise

FROM Employees

* JOINS

table\_left

id, FName, LName,

table\_right

id, Department, Age

SELECT

table\_left.FName, table\_left.LName,

table\_right.Department, table\_right.Age

FROM

table\_left LEFT JOIN table\_right

ON

table\_left.id = table\_right.id

* Length of string

SELECT

FName, LENGTH(FName) AS alphabets\_in\_FName,

LName, LENGTH(LName) AS alphabets\_in\_LName

FROM employees

* SUBSTR (Extract date, month and year)

date

31-12-1991

SELECT

event,

SUBSTR(date, 1, 2) AS date,

SUBSTR(date, 4,2) AS Month,

SUBSTR(date, 7, 4) AS Year

FROM events\_data

* INSTR identifies location of a given alphabet in a string

SELECT INSTR (“mangal@gmail.com”, “@”)

7

* TRIM

SELECT TRIM(“ Mangaljit ”)

SELECT TRIM (“…Mangaljit…”, “.”)

* Common Table Expression (CTE)

WITH temp\_table AS

(SELECT region, AVG(pricem) AS avg\_price

FROM album.clean\_combined\_toronto\_property\_data

GROUP BY region)

SELECT \* FROM temp\_table

WHERE

avg\_price = (SELECT MAX(avg\_price) FROM temp\_table);

* ROW\_NUMBER()

This adds a row number 1,2,3… in a new column

SELECT \*,

ROW\_NUMBER() OVER() index

FROM data

* PARTITION BY

Select second highest price among each region

WITH temp\_table AS

(SELECT region, pricem,

ROW\_NUMBER() OVER(PARTITION BY region ORDER BY pricem DESC) AS price\_rank

FROM album.clean\_combined\_toronto\_property\_data)

SELECT \* FROM temp\_table

WHERE price\_rank = 2;

Add average price of region in new column without reducing the number of rows

SELECT \*,

AVG(price), OVER(PARTITION BY region) AS average\_price

FROM data

* F
* F
* F
* F
* F
* F
* F
* F
* F
* F
* F
* F
* Ff