```sql
--------------------------------------------
--Selecting All Columns
--------------------------------------------

SELECT * FROM customer;

--------------------------------------------
--Selecting Specific Columns
--------------------------------------------

SELECT customer_id, first_name, last_name
FROM customer;

SELECT title, release_year, length
FROM film;

--------------------------------------------
--Arithmetic Expressions
--------------------------------------------
SELECT 8 * 5 + 4,
      120 / 4,
      current_date - 2;

SELECT title,
    length / 60,
    2021 - release_year
FROM film;

--------------------------------------------
--Concatenation Operator
--------------------------------------------

SELECT country_id || ': ' || country
FROM country;

SELECT customer_id || ': ' || first_name || ' ' || last_name
FROM customer;

--------------------------------------------
--Escape Character - E
--------------------------------------------

SELECT 'Computer Engineer';

SELECT 'I''m a Computer Engineer';

SELECT E'I\'m a computer engineer';

SELECT customer_id || E'\'s: '
  || first_name || ' '
  || last_name
FROM customer;

--------------------------------------------
--Escape Character - $$
```

```
--------------------------------------------

SELECT $$'I'm a Computer Engineer'$$;

SELECT $msg$'I'm a string constant that contains a backslash \'$msg$;

--------------------------------------------
--Column Aliases
--------------------------------------------

SELECT first_name || ' ' || last_name as full_name
FROM customer;

SELECT title,
    length as length_minute,
    round(length::decimal / 60, 2) as length_hour,
    (2021 - release_year) pass_year
FROM film;

--------------------------------------------
--Order By - Single Column
--------------------------------------------

SELECT * FROM film
ORDER BY title

SELECT district, address, city_id
FROM address
ORDER BY district

SELECT first_name, last_name, email, address_id
FROM customer
ORDER BY first_name DESC

--------------------------------------------
--Order By - Multiple Column
--------------------------------------------

SELECT rating, length, title, rental_rate
FROM film
ORDER BY rating, length DESC;

SELECT first_name, last_name,
       email, address_id
FROM customer
ORDER BY first_name, last_name

--------------------------------------------
--Order By - Column Order
--------------------------------------------

SELECT *
FROM customer
ORDER BY 6
```

```
SELECT first_name, last_name,
       email, hire_date,
       job_id, salary
FROM employees
ORDER BY 5, 6 DESC

--------------------------------------------
--Order By - Nulls First, Last
--------------------------------------------

SELECT * FROM employees
ORDER BY phone_number NULLS LAST

SELECT * FROM employees
ORDER BY phone_number NULLS FIRST

--------------------------------------------
--Duplicate Rows - DISTINCT
--------------------------------------------

SELECT DISTINCT special_features
FROM film

--------------------------------------------
--Duplicate Rows - DISTINCT ON
--------------------------------------------

SELECT DISTINCT bcolor, fcolor
FROM distinct_demo
ORDER BY bcolor, fcolor;

SELECT DISTINCT ON (bcolor) bcolor,  fcolor
FROM distinct_demo
ORDER BY bcolor, fcolor;

SELECT DISTINCT ON (film_id) film_id, store_id
FROM inventory
ORDER BY film_id, store_id;
```

```
-----------------------------------------------------
--EXERCISE ANSWERS
-----------------------------------------------------

--------------------------------------------
--Simple Query
--------------------------------------------

SELECT first_name ||'.'|| last_name ||'@dbhunter.com',
    salary * 1.35
FROM employees;

--------------------------------------------
--DISTINCT
--------------------------------------------

SELECT DISTINCT rating
FROM film
ORDER BY 1 DESC
```

```sql
--------------------------------------------
--Comparison Operators
--------------------------------------------

SELECT * FROM actor
WHERE first_name = 'Penelope';

SELECT * FROM customer
WHERE store_id = 1;

SELECT * FROM payment
WHERE staff_id <> 2;

SELECT * FROM film
WHERE length >= 100;

--------------------------------------------
--Logical Operators
--------------------------------------------

SELECT * FROM film
WHERE length > 100
      AND rental_duration < 5;

SELECT * FROM address
WHERE district = 'Adana'
       AND city_id = 5;

SELECT * FROM payment
WHERE staff_id = 2
      OR amount > 5;

SELECT * FROM employees
WHERE department_id = 1
      OR salary < 3000

SELECT * FROM employees
WHERE job_id = 9
       AND salary > 5000;

--------------------------------------------
--BETWEEN Operator
--------------------------------------------

SELECT * FROM film
WHERE length BETWEEN 100 AND 120;

SELECT * FROM employees
WHERE salary BETWEEN 5000 AND 10000;

SELECT * FROM employees
WHERE hire_date BETWEEN '2017-01-01' and '2017-12-31'
ORDER BY hire_date;

SELECT * FROM employees
WHERE first_name BETWEEN 'A' and 'E'
ORDER BY first_name;
```

```sql
--------------------------------------------
--IN Operator
--------------------------------------------

SELECT first_name, last_name
FROM customer
WHERE first_name IN ('Leslie', 'Kelly', 'Tracy');

SELECT *
FROM film
WHERE rating IN ('R', 'G')

SELECT *
FROM address
WHERE district IN ('Texas', 'Nantou', 'Moskova');

SELECT * FROM customer
WHERE address_id IN (10, 20, 30);

SELECT * FROM customer
WHERE address_id = 10
      OR address_id = 20
      OR address_id = 30

--------------------------------------------
--LIKE Operator
--------------------------------------------

SELECT first_name, last_name
FROM customer
WHERE first_name LIKE 'Ann%';

SELECT *
FROM customer
WHERE first_name LIKE 'B%';

SELECT * FROM film
WHERE title LIKE '%r';

SELECT * FROM film
WHERE title LIKE '%u_';

SELECT * FROM film
WHERE title LIKE '%s%';

SELECT * FROM film
WHERE title LIKE '%a%v%';

--------------------------------------------
--ILIKE Operator
--------------------------------------------

SELECT first_name, last_name
FROM customer
WHERE first_name ILIKE 'CAR%';
```

```
--------------------------------------------
--NOT Operator
--------------------------------------------

SELECT * FROM employees
WHERE manager_id NOT IN (100, 108, 114)

SELECT * FROM employees
WHERE first_name NOT LIKE 'A%'

--------------------------------------------
--IS NULL Operator
--------------------------------------------

SELECT * FROM employees
WHERE phone_number IS NULL;

SELECT * FROM employees
WHERE phone_number IS NOT NULL;

--------------------------------------------
--LIMIT Clause
--------------------------------------------

SELECT * FROM actor
ORDER BY first_name, last_name
LIMIT 5;

SELECT film_id, title
FROM film
ORDER BY film_id
LIMIT 8 OFFSET 4;

--------------------------------------------
--FETCH Clause
--------------------------------------------

SELECT film_id, title
FROM film
ORDER BY title
FETCH FIRST ROW ONLY;

SELECT film_id, title
FROM film
ORDER BY title
FETCH FIRST 1 ROW ONLY;

SELECT film_id, title
FROM film
ORDER BY title
OFFSET 5 ROWS
FETCH FIRST 6 ROW ONLY;

SELECT film_id, title
FROM film
ORDER BY title
LIMIT 6 OFFSET 5;
```

```sql
-----------------------------------------------------
--EXERCISE ANSWERS
-----------------------------------------------------

--------------------------------------------
--Between Operator
--------------------------------------------

SELECT * FROM staff
WHERE '2022-05-01' BETWEEN hire_date AND departure_date

--------------------------------------------
--LIKE Operator
--------------------------------------------

SELECT first_name, last_name
FROM customer
WHERE first_name LIKE '_her%';

--------------------------------------------
--NOT Operator
--------------------------------------------

SELECT first_name, last_name, salary, job_id
FROM employees
WHERE
    NOT salary BETWEEN 4000 and 7000
    AND NOT job_id = 16
```

```sql
--------------------------------------------
--String Functions - Letter Case
--------------------------------------------

SELECT title,
      LOWER(title) lo_title,
      UPPER(title) up_title,
      INITCAP(title) ic_title
FROM film;

SELECT  *
FROM film
WHERE LOWER(title) LIKE '%trip%'

--------------------------------------------
--String Functions - Character Processing
--------------------------------------------

SELECT first_name, last_name,
      CONCAT(first_name, ' ', last_name) as f1,
      CONCAT_WS(',', first_name, last_name) as f2,
      LEFT(first_name, 1) as f3,
      RIGHT(first_name, 1) as f4,
      LPAD(phone_number, 14, '00') as f6,
      RPAD(phone_number, 14, '00') as f7,
FROM employees;

SELECT first_name, last_name,
      LENGTH(last_name) as f5,
      REPLACE(first_name, 'e', ' * ') as f8,
      SPLIT_PART(hire_date::varchar, '-', 2) as f9,
      SUBSTRING(last_name, 2, 3) as f10,
      POSITION('a' in first_name) as f11,
      REVERSE(first_name) as f12
FROM employees;

--------------------------------------------
--Math Functions
--------------------------------------------

SELECT ROUND(14.45),
       CEIL(14.45),
       FLOOR(14.45),
       ABS(-5.78),
       POWER(4,3),
       SIGN(-5),
       TRUNC(4836.98);

SELECT payment_id, amount,
      (amount * 0.45) as percent_amount,
      ROUND(amount * 0.45) as f_round,
      CEIL(amount * 0.45) as f_ceil,
      FLOOR(amount * 0.45) as f_floor,
      TRUNC(amount * 0.45) as f_trunc,
      MOD(amount, 5) as f_mod
FROM payment
LIMIT 10;
```

```sql
--------------------------------------------
--Math Functions-Random
--------------------------------------------

SELECT random();

SELECT random() * 100 + 1 as ran_num;

SELECT floor(random() * 100 + 1)::int as ran_num;

SELECT floor(random() * (high - low + 1) + low)::int as ran_num;

SELECT floor(random() * (200 - 100 + 1) + 100)::int as ran_num
FROM generate_series(1, 10);

--------------------------------------------
--Date Functions
--------------------------------------------

SELECT CURRENT_DATE,
       CURRENT_TIME,
       LOCALTIME,
       NOW(),
       TIMEOFDAY();

SELECT first_name,
       last_name,
       hire_date,
       AGE(hire_date) as age_of_hire
FROM employees;

--------------------------------------------
--Date Functions - DATE_PART
--------------------------------------------

SELECT CURRENT_DATE,
      DATE_PART('century', CURRENT_DATE) as century_,
      DATE_PART('quarter', CURRENT_DATE) as quarter_,
      DATE_PART('decade', CURRENT_DATE) as decade_,
      DATE_PART('year', CURRENT_DATE) as year_,
      DATE_PART('month', CURRENT_DATE) as month_,
      DATE_PART('day', CURRENT_DATE) as day_,
      DATE_PART('hour', CURRENT_DATE) as hour_,
      DATE_PART('minute', CURRENT_DATE) as minute_,
      DATE_PART('dow', CURRENT_DATE) as dow_,
      DATE_PART('doy', CURRENT_DATE) as doy_,
      DATE_PART('timezone', CURRENT_TIME) as timezone_;
```

```sql
--------------------------------------------
--Date Functions - EXTRACT
--------------------------------------------

SELECT CURRENT_DATE,
      EXTRACT(CENTURY FROM CURRENT_DATE) as century_,
      EXTRACT(QUARTER FROM CURRENT_DATE) as quarter_,
      EXTRACT(DECADE FROM CURRENT_DATE) as decade_,
      EXTRACT(YEAR FROM CURRENT_DATE) as year_,
      EXTRACT(MONTH FROM CURRENT_DATE) as month_,
      EXTRACT(DAY FROM CURRENT_DATE) as day_,
      EXTRACT(HOUR FROM CURRENT_TIME) as hour_,
      EXTRACT(MINUTE FROM CURRENT_TIME) as minute_,
      EXTRACT(DOW FROM CURRENT_DATE) as dow_,
      EXTRACT(DOY FROM CURRENT_DATE) as doy_,
      EXTRACT(TIMEZONE FROM CURRENT_TIME) as timezone_;

--------------------------------------------
--Date Functions - DATE_TRUNC
--------------------------------------------

SELECT
       DATE_TRUNC('quarter', TIMESTAMP '2023-04-17 06:12:38') as quarter_,
       DATE_TRUNC('year',    TIMESTAMP '2023-04-17 06:12:38') as year_,
       DATE_TRUNC('month',   TIMESTAMP '2023-04-17 06:12:38') as month_,
       DATE_TRUNC('day',     TIMESTAMP '2023-04-17 06:12:38') as day_,
       DATE_TRUNC('hour',    TIMESTAMP '2023-04-17 06:12:38') as hour_,
       DATE_TRUNC('minute',  TIMESTAMP '2023-04-17 06:12:38') as minute_,
       DATE_TRUNC('second',  TIMESTAMP '2023-04-17 06:12:38') as second_

--------------------------------------------
--Conversion Functions - TO_DATE
--------------------------------------------

SELECT TO_DATE('20230405', 'YYYYMMDD') TD1,
       TO_DATE('2023 APRIL 05', 'YYYY MONTH DD') TD2,
       TO_DATE('2023 april 05', 'YYYY month DD') TD3,
       TO_DATE('2023 APR 05', 'YYYY MON DD') TD4,
       TO_DATE('2023 240', 'YYYY DDD') TD5,
       TO_DATE('February 08, 2023', 'Month DD, YYYY') TD6;

--------------------------------------------
--Conversion Functions - TO_TIMESTAMP
--------------------------------------------

SELECT TO_TIMESTAMP('2021-05-30 08:40:30', 'YYYY-MM-DD HH:MI:SS') TT1,
       TO_TIMESTAMP('05.30.2021 20:40:30', 'MM.DD.YYYY HH24:MI:SS') TT2,
       TO_TIMESTAMP('2022/25/08 08:40', 'YYYY/DD/MM HH:MI') TT3,
       TO_TIMESTAMP('11 30 99 12:40', 'MM DD YY HH:MI') TT4,
       TO_TIMESTAMP('09 07 19 10:35', 'MM DD YY HH:MI') TT5,
       TO_TIMESTAMP('2022 OCT 15 07:21:11', 'YYYY MON DD HH:MI:SS') TT6;
```

```sql
--------------------------------------------
--Conversion Functions — TO_NUMBER
--------------------------------------------

SELECT TO_NUMBER('1210.73', '9999.99') TN1,
       TO_NUMBER('1,210.73', '9G999.99') TN2,
       TO_NUMBER('$1,210.73', 'L9G999.99') TN3,
       TO_NUMBER('$1,210.73', 'L9G999') TN4,
       TO_NUMBER('-12.345,6', '99G999D9S') TN5,
       TO_NUMBER('$1.234.567,89', 'L9G999g999,99') TN6,
       TO_NUMBER('00005469', '9999999999') TN7,
       '00005469'::integer TN8,
       CAST('00005469' as integer) TN9;

--------------------------------------------
--Conversion Functions — TO_CHAR
--------------------------------------------

SELECT payment_id, payment_date, amount,
       TO_CHAR(payment_date, 'HH24:MI:SS')            as TC1,
       TO_CHAR(payment_date, 'MON-DD-YYYY HH12:MI PM') as TC2,
       TO_CHAR(payment_date, 'DD.MM.YYYY HH24:MI')    as TC3,
       TO_CHAR(payment_date, 'MON-DAY-YYYY HH12:MI')  as TC4,
       TO_CHAR(payment_date, 'Month DD, YYYY')        as TC5,
       TO_CHAR(payment_date, 'YYYYMMDD')              as TC6,
       TO_CHAR(amount,       '99D99')                 as TC7
FROM payment;

--------------------------------------------
--CAST Function
--------------------------------------------

SELECT
    CAST ('100' as INTEGER) as cast1,
    CAST ('2021-01-01' as DATE) as cast2,
    CAST ('15-OCT-2022' as DATE) as cast3,
    CAST ('10.25' as DOUBLE PRECISION) as cast4,
    CAST ('true' as BOOLEAN) as cast5,
    CAST ('false' as BOOLEAN) as cast6,
    CAST ('T' as BOOLEAN) as cast7,
    CAST ('F' as BOOLEAN) as cast8;

SELECT
    '100'::INTEGER  as cast1,
    '01-OCT-2015'::DATE as cast2,
    598::VARCHAR as cast3,
    '2019-06-15 14:30:20'::timestamp as cast4,
    '15 minute'::interval as cast5,
    '2 hour'::interval as cast6,
    '1 day'::interval as cast7,
    '2 week'::interval as cast8,
    '3 month'::interval as cast9;
```

```sql
SELECT
    CAST ('2 year 5 months 3 days' AS INTERVAL),
    CAST (2800 AS MONEY),
    CAST (CURRENT_DATE AS TEXT);

SELECT
    date_value_str,
    CAST (date_value_str AS DATE)
FROM date_demo;

--------------------------------------------
--Arithmetic Operations with Dates
--------------------------------------------

SELECT current_date, current_time,
       current_date + 10 as F1,
       current_date - 5 as F2,
       current_date - TO_DATE('01012022', 'DDMMYYYY') as F3,
       current_time + INTERVAL '2 hour' as F4,
       NOW() - INTERVAL '1 year 2 months 3 hours 20 minutes' as F5;

--------------------------------------------
--COLAESCE Function
--------------------------------------------

SELECT
    COALESCE(1, 2, 3) C1,
    COALESCE(null, 2, 3) C2,
    COALESCE(null, null, 3) C2;

SELECT phone_number ,
       COALESCE(phone_number, 'No phone number')
FROM employees;

--------------------------------------------
--NULLIF Function
--------------------------------------------

SELECT
    NULLIF(1, 1) N1,
    NULLIF(1, 2) N2,
    NULLIF('A', 'B') N3;

SELECT b.*,
       NULLIF(current_year, previous_year) as budget
FROM budgets b;
```

```sql
--------------------------------------------
--CASE Expression
--------------------------------------------

SELECT title, length,
    CASE
        WHEN length >= 0   AND length <= 50 THEN 'Short length'
        WHEN length >= 51  AND length <= 120 THEN 'Medium length'
        WHEN length > 120 THEN 'Long length'
    END duration
FROM film
ORDER BY title;

SELECT title, rating,
    CASE rating
        WHEN 'G' THEN 'General Audiences'
        WHEN 'PG' THEN 'Parental Guidance Suggested'
        WHEN 'PG-13' THEN 'Parents Strongly Cautioned'
        WHEN 'R' THEN 'Restricted'
        WHEN 'NC-17' THEN 'Adults Only'
    END rating_description
FROM film
ORDER BY title;

SELECT first_name, last_name, job_id,
    CASE
        WHEN job_id in (2,7,10,14,15,19) THEN 'Manager Positions'
        WHEN job_id in (13,17,18) THEN 'Clerk Positions'
        ELSE 'Other Positions'
    END position_type
FROM employees;

--------------------------------------------
--Nested Functions
--------------------------------------------

SELECT first_name, last_name,
    LENGTH(CONCAT(first_name, last_name)) as length_name,
    CONCAT(SUBSTRING(first_name, 1, 2), '.', SUBSTRING(last_name, 1, 2), '.') as
name
FROM employees

SELECT salesman_id,
  COALESCE(
          CAST(
                NULLIF(current_year, previous_year)
                as Varchar),
          'Same as last year') as budget
FROM budgets
WHERE current_year IS NOT NULL;
```

```
-----------------------------------------------------
--EXERCISE ANSWERS
-----------------------------------------------------

--------------------------------------------
--String Functions - Letter Case
--------------------------------------------

select
    upper(title) as title_new,
    lower(description) as description_new
from film
where
    lower(description) like '%drama%'
    and lower(description) like '%australia%'

--------------------------------------------
--String Functions - Character Processing
--------------------------------------------

--Exercise-1:

SELECT title, description FROM film
WHERE description like '%Hunter%'

SELECT title, description FROM film
WHERE initcap(description) like '%Hunter%'

SELECT title, description FROM film
WHERE position('Hunter' in description) > 0


--Exercise-2:

SELECT first_name, last_name,
       CONCAT(first_name, ' ', last_name, ' ', email) as f1,
       CONCAT_WS(' ', first_name, last_name, email) as f2,
       first_name || ' ' || last_name || ' ' || email as f3
FROM employees;

--------------------------------------------
--Date Functions
--------------------------------------------

SELECT
    'Quarter is: ' || DATE_PART('quarter', CURRENT_DATE) || ', '
    'Year is: '    || DATE_PART('year', CURRENT_DATE)    || ', '
    'Month is: '   || DATE_PART('month', CURRENT_DATE)   || ', '
    'Doy is: '     || DATE_PART('doy', CURRENT_DATE)     || '.'

SELECT
    'Quarter is: ' || EXTRACT(quarter from CURRENT_DATE) || ', '
    'Year is: '    || EXTRACT(year from CURRENT_DATE)    || ', '
    'Month is: '   || EXTRACT(month from CURRENT_DATE)   || ', '
    'Doy is: '     || EXTRACT(doy from CURRENT_DATE)     || '.'
```

```sql
---------------------------------------------
--Conversion Functions - TO_CHAR
---------------------------------------------

SELECT customer_id
       || ' paid: ' ||
       TO_CHAR(amount,'$99D99')
       || ' at ' ||
       TO_CHAR(payment_date, 'HH24:MI:SS')
       || ' on ' ||
       TO_CHAR(payment_date, 'Mon-DD-YYYY') as p_info
FROM payment
ORDER BY rental_id
LIMIT 5;

---------------------------------------------
--COLAESCE Function
---------------------------------------------

SELECT
    brand, price, discount,
    (price - COALESCE(discount,0)) as net_price
FROM cars;

---------------------------------------------
--NULLIF Function
---------------------------------------------

SELECT b.*,
       COALESCE(current_year, previous_year) as budget1,
       COALESCE(NULLIF(current_year,NULL), previous_year) as budget2
FROM budgets b;

---------------------------------------------
--CASE Expression
---------------------------------------------

select first_name, last_name, hire_date, salary,
  case
    when date_part('year', hire_date) between 2018 and 2020 then salary * 1.25
    when date_part('year', hire_date) < 2018 then salary * 1.35
    when date_part('year', hire_date) > 2020 then salary * 1.15
  end as new_salary
from employees
```

```
--------------------------------------------
--AVG Function
--------------------------------------------

SELECT
    AVG(length) as avg_length,
    AVG(rental_duration) as avg_rental_duration,
    AVG(replacement_cost) as avg_replacement_cost
FROM film;

SELECT
    AVG(length) as avg1,
    AVG(DISTINCT length) as avg2
FROM film;

--------------------------------------------
--SUM Function
--------------------------------------------

SELECT
    SUM(length) as sum_length,
    SUM(rental_duration) as sum_rental_duration,
    SUM(replacement_cost) as sum_replacement_cost
FROM film;

SELECT
    SUM(salary) as sum_manager_sal,
    ROUND(AVG(salary),2) as avg_manager_sal
FROM employees
WHERE job_id in (2,7,10,14,15,19) -manager positios;


--------------------------------------------
--COUNT Function
--------------------------------------------

SELECT COUNT(*) FROM actor;

SELECT COUNT(*) FROM film
WHERE rating = 'G';

SELECT COUNT(*) FROM payment
WHERE customer_id = 341;

SELECT COUNT(phone_number) FROM employees;

SELECT COUNT(discount) FROM cars;

SELECT COUNT(distinct rating) FROM film;

SELECT COUNT(distinct job_id) FROM employees;
```

```sql
--------------------------------------------
--MIN-MAX Functions
--------------------------------------------

SELECT
    MIN(salary) min_salary,
    MAX(salary) max_salary,
    MIN(hire_date) min_hire_date,
    MAX(hire_date) max_hire_date
FROM employees;

SELECT
    MIN(length) min_length,
    MAX(length) max_length,
    MAX(replacement_cost) max_replacement_cost,
    MAX(rental_duration) - MIN(rental_duration) dif_rental_duration
FROM film;

--------------------------------------------
--GROUP BY Clause
--------------------------------------------

SELECT DISTINCT rating
FROM film
ORDER BY rating;

SELECT rating
FROM film
GROUP BY rating
ORDER BY rating;

SELECT
    customer_id,
    SUM (amount)
FROM payment
GROUP BY customer_id;

SELECT
    rating,
    SUM(length) sum_length,
    SUM(rental_duration) sum_rental_duration,
    SUM(replacement_cost) sum_replacement_cost
FROM film
GROUP BY rating
ORDER BY rating;

SELECT
    job_id,
    COUNT(*) numberof_emp,
    MIN(salary) min_salary,
    MAX(salary) max_salary
FROM employees
GROUP BY job_id
ORDER BY 1;
```

```sql
SELECT
    rating, special_features,
    COUNT(*) numberof_films
FROM film
GROUP BY rating, special_features
ORDER BY rating, special_features;

SELECT
    department_id, manager_id,
    COUNT(*) numberof_emp
FROM employees
GROUP BY department_id, manager_id
ORDER BY 1, 2;


--------------------------------------------
--Having Clause
--------------------------------------------

SELECT
    customer_id,
    SUM(amount) sum_amount
FROM payment
GROUP BY customer_id
HAVING SUM(amount) > 150

SELECT
    department_id,
    COUNT(*) numberof_emps
FROM employees
GROUP BY department_id
HAVING COUNT(*) > 5
```

```
-----------------------------------------------------
--EXERCISE ANSWERS
-----------------------------------------------------

--------------------------------------------
--AVG Function
--------------------------------------------

SELECT
      ROUND(AVG(length), 2) as avg_length,
      ROUND(AVG(rental_duration), 2) as avg_rental_duration,
      ROUND(AVG(replacement_cost), 2) as avg_replacement_cost
FROM film;

--------------------------------------------
--SUM Function
--------------------------------------------

SELECT SUM(return_date - rental_date)
FROM rental;

SELECT
    return_date, rental_date,
    return_date::date - rental_date::date
FROM rental;

--------------------------------------------
--COUNT Function
--------------------------------------------

select
    count(*) as number_of_rows,
    count(postal_code) as non_null_postal_code,
    count(distinct district) different_districts
from address;

--------------------------------------------
--MIN-MAX Functions
--------------------------------------------

SELECT
    MIN(length(CONCAT(first_name, last_name))) min_numberof_letters,
    MAX(length(CONCAT(first_name, last_name))) max_numberof_letters,
    ROUND(AVG(length(CONCAT(first_name, last_name))),2) avg_numberof_letters
FROM employees;
```

```
--------------------------------------------
--GROUP BY Clause
--------------------------------------------

--Exercise-1

select customer_id,
    count(*) as rental_count,
    min(rental_date)::date as first_rental_date,
    max(rental_date)::date as first_rental_date
from rental
group by customer_id

--Exercise-2

SELECT
    SUM(CASE rating WHEN 'NC-17' THEN 1 ELSE 0 end) numberof_NC17,
    SUM(CASE rating WHEN 'PG' THEN 1 ELSE 0 end) numberof_PG,
    SUM(CASE rating WHEN 'G' THEN 1 ELSE 0 end) numberof_G,
    SUM(CASE rating WHEN 'PG-13' THEN 1 ELSE 0 end) numberof_PG13,
    SUM(CASE rating WHEN 'R' THEN 1 ELSE 0 end) numberof_R
FROM film;


--------------------------------------------
--Having Clause
--------------------------------------------

--Exercise-1

SELECT customer_id, COUNT (*)
FROM rental
GROUP BY customer_id
HAVING COUNT (*) > 35;

--Exercise-2

SELECT department_id,
    COUNT(*) numberof_emps,
    CASE
        WHEN COUNT(*) <= 3 THEN 'Small Room'
        WHEN COUNT(*) BETWEEN 4 AND 6 THEN 'Middle Room'
        WHEN COUNT(*) > 6 THEN 'Big Room'
    END AS room_type
FROM employees
GROUP BY department_id
ORDER BY 2
```

```sql
--------------------------------------------
--INNER JOIN
--------------------------------------------

SELECT
    id_a, fruit_a,
    id_b, fruit_b
FROM basket_a
INNER JOIN basket_b
    ON fruit_a = fruit_b

SELECT
    c.customer_id,
    first_name, last_name,
    amount, payment_date
FROM customer c
INNER JOIN payment p
    ON p.customer_id = c.customer_id
ORDER BY payment_date;

--------------------------------------------
--JOIN with USING
--------------------------------------------

SELECT customer_id,
    first_name, last_name,
    amount, payment_date
FROM customer
INNER JOIN payment USING(customer_id)
ORDER BY payment_date;

--------------------------------------------
--JOIN with Classical Way
--------------------------------------------

SELECT c.customer_id,
    first_name, last_name,
    amount, payment_date
FROM customer c, payment p
WHERE p.customer_id = c.customer_id
ORDER BY payment_date;

--------------------------------------------
--LEFT JOIN
--------------------------------------------

SELECT
    a.film_id, a.title, b.inventory_id
FROM film a
LEFT JOIN inventory b ON b.film_id = a.film_id
ORDER BY title;

SELECT
    a.film_id, a.title, b.inventory_id
FROM film a
LEFT JOIN inventory b USING (film_id)
ORDER BY title;
```

```
SELECT
    e.first_name, e.last_name,
    d.first_name child_first_name,
d.last_name child_last_name
FROM employees e
LEFT JOIN dependents d
     ON d.employee_id = e.employee_id

--------------------------------------------
--RIGHT JOIN
--------------------------------------------

SELECT
   m.movie_id, mr.movie_id,
   review, title
FROM movies m
RIGHT JOIN movie_reviews mr
   ON mr.movie_id = m.movie_id;

SELECT
    id_a, fruit_a,
    id_b, fruit_b
FROM basket_a
RIGHT JOIN basket_b
    ON fruit_a = fruit_b;

--------------------------------------------
--FULL OUTER JOIN
--------------------------------------------

SELECT
   m.movie_id, mr.movie_id,
   review, title
FROM movies m
FULL OUTER JOIN movie_reviews mr
   ON mr.movie_id = m.movie_id;

SELECT
    id_a, fruit_a,
    id_b, fruit_b
FROM basket_a
FULL JOIN basket_b
    ON fruit_a = fruit_b;

--------------------------------------------
--SELF JOIN
--------------------------------------------

SELECT
    m.first_name || ' ' || m.last_name manager,
    e.first_name || ' ' || e.last_name employee
FROM employees e
INNER JOIN employees m
  ON e.manager_id = m.employee_id
ORDER BY manager
```

```sql
SELECT
    f1.title, f2.title, f1.length
FROM film f1
INNER JOIN film f2
    ON f1.film_id <> f2.film_id
    AND f1.length = f2.length

--------------------------------------------
--CROSS JOIN
--------------------------------------------

SELECT brand, color
FROM cars
CROSS JOIN colors;

SELECT brand, color
FROM cars
INNER JOIN colors ON true;

SELECT brand, color
FROM cars, colors;

--------------------------------------------
--NATURAL JOIN
--------------------------------------------

SELECT
    e.first_name || ' ' || e.last_name employee,
    d.department_name
FROM employees e
NATURAL JOIN departments d;

SELECT
    e.first_name || ' ' || e.last_name employee,
    d.department_name
FROM employees e
INNER JOIN departments d USING(department_id);

--------------------------------------------
--NON EQUAL JOIN
--------------------------------------------

SELECT
    first_name, last_name, salary,
    min_salary, max_salary, job_title
FROM employees e, jobs j
WHERE e.salary BETWEEN j.min_salary AND j.max_salary;
```

```sql
-----------------------------------------------------
--EXERCISE ANSWERS
-----------------------------------------------------

--------------------------------------------
--INNER JOIN
--------------------------------------------

--Exercise-1

SELECT first_name, last_name, d.department_name
FROM employees e, departments d
WHERE e.department_id = d.department_id;

SELECT first_name, last_name, d.department_name
FROM employees e INNER JOIN departments d
ON e.department_id = d.department_id;

--Exercise-2

SELECT
    c.first_name as customer_first_name,
    c.last_name as customer_last_name,
    p.amount,
    s.first_name as staff_first_name,
    s.last_name as staff_last_name
FROM customer c
INNER JOIN payment p
    ON p.customer_id = c.customer_id
INNER JOIN staff s
    ON p.staff_id = s.staff_id
WHERE c.customer_id = 341

SELECT
    c.first_name as customer_first_name,
    c.last_name as customer_last_name,
    p.amount,
    s.first_name as staff_first_name,
    s.last_name as staff_last_name
FROM customer c, payment p, staff s
WHERE p.customer_id = c.customer_id
     and p.staff_id = s.staff_id
     and c.customer_id = 341;
```

```
--------------------------------------------
--SELF JOIN
--------------------------------------------

select
    c1.first_name as c1_first_name,
    c1.last_name  as c1_last_name,
    c2.first_name as c2_first_name,
    c2.last_name  as c2_last_name,
    ct.city
from
    customer c1, customer c2,
    address a1, address a2,
    city ct
where c1.customer_id <> c2.customer_id
    and c1.address_id = a1.address_id
    and c2.address_id = a2.address_id
    and a1.address_id <> a2.address_id
    and a1.city_id = a2.city_id
    and a1.city_id = ct.city_id

--------------------------------------------
--CROSS JOIN
--------------------------------------------

SELECT c1.brand, c2.type_name, c3.color
FROM cars c1, car_types c2, colors c3
ORDER BY 1,2,3

SELECT c1.brand, c2.type_name, c3.color
FROM cars c1
CROSS JOIN car_types c2
CROSS JOIN colors c3
ORDER BY 1,2,3
```

```sql
--------------------------------------------
--UNION Operator
--------------------------------------------

SELECT fruit_a FROM basket_a
UNION
SELECT fruit_b FROM basket_b

select first_name, last_name, job_id, salary
from employees
where salary between 8000 and 12000
UNION
select first_name, last_name, job_id, salary
from employees
where job_id in (2,7,10,14,15,19)
order by 1,2;

--------------------------------------------
--UNION ALL Operator
--------------------------------------------

select first_name, last_name, job_id, salary
from employees
where salary between 8000 and 12000
UNION ALL
select first_name, last_name, job_id, salary
from employees
where job_id in (2,7,10,14,15,19)
order by 1,2;

SELECT title, length, rating
FROM film
WHERE length < 50
UNION ALL
SELECT title, length, rating
FROM film
WHERE rating = 'PG'
UNION ALL
SELECT title, length, rating
FROM film
WHERE rating = 'R'
ORDER BY title;

--------------------------------------------
--INTERSECT Operator
--------------------------------------------

select first_name, last_name, job_id, salary
from employees
where salary between 8000 and 12000
INTERSECT
select first_name, last_name, job_id, salary
from employees
where job_id in (2,7,10,14,15,19)
order by 1,2;
```

```
select title, description, length, rating
from film
where rating = 'G'
intersect
select title, description, length, rating
from film
where length > 170

--------------------------------------------
--EXCEPT Operator
--------------------------------------------
SELECT fruit_a FROM basket_a
EXCEPT
SELECT fruit_b FROM basket_b

select first_name, last_name, job_id, salary
from employees
where salary between 8000 and 12000
EXCEPT
select first_name, last_name, job_id, salary
from employees
where job_id in (2,7,10,14,15,19)
order by 1,2;

--------------------------------------------
--Combining Different Operators
--------------------------------------------

(
    select title, description, length, rating
    from film
    where length > 170
    UNION
    select title, description, length, rating
    from film
    where rating = 'G'
)
EXCEPT
select title, description, length, rating
from film
where rating = 'G'
  and rental_duration < 5
```

```sql
-----------------------------------------------------
--EXERCISE ANSWERS
-----------------------------------------------------

--------------------------------------------
--UNION Operator
--------------------------------------------

select title, length, rating
from film
where rating = 'PG'
UNION
select title, length, rating
from film f
where f.special_features::varchar like '%Trailers%'

--------------------------------------------
--UNION ALL Operator
--------------------------------------------

select
  'Country' as place_type,
  country_id as place_id,
  country  as place_name
from country
union all
select
  'City' as place_type,
  city_id as place_id,
  city as place_name
from city
union all
select
  'Address' as place_type,
  address_id as place_id,
  address  as place_name
from address

--------------------------------------------
--EXCEPT Operator
--------------------------------------------

select film_id, title
from film
except
select distinct f.film_id, title
from inventory i
inner join film f
      on f.film_id = i.film_id

select film_id, title
from film
where film_id not in
      (
            select distinct film_id
            from inventory i
      )
```

```sql
select film_id, title
from film f
where not exists
      (
            select 1
            from inventory i
            where f.film_id = i.film_id
      )
```

```sql
--------------------------------------------
--Single-Row Subqueries - Where
--------------------------------------------

SELECT * FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees)

SELECT title, length, rating
FROM film
WHERE length = (SELECT length FROM film WHERE title = 'Godfather Diary')

SELECT title, length, rating
FROM film
WHERE (length, rating) =
      (SELECT length, rating FROM film WHERE title = 'Home Pity')

--------------------------------------------
--Single-Row Subqueries - Column
--------------------------------------------

SELECT
    first_name, last_name,
    department_name
FROM employees e
INNER JOIN departments d
     ON d.department_id = e.department_id;

SELECT
    first_name,last_name,
    (
        SELECT department_name FROM departments d
        WHERE d.department_id = e.department_id
    ) dep_name
FROM employees e;

--------------------------------------------
--Single-Row Subqueries - Having
--------------------------------------------

SELECT department_id,
       COUNT(*) as emp_num
FROM employees e
GROUP BY department_id
HAVING COUNT(*) >
    (
       SELECT COUNT(*)
       FROM employees
       WHERE department_id = 6
    )
```

```
--------------------------------------------
--Multiple-Row Subqueries - FROM
--------------------------------------------

select
  e.first_name, e.last_name,
  dl.department_name, dl.city, dl.state_province
from employees e,
  (
    select
      d.department_id, d.department_name,
      l.city, l.state_province
    from departments d, locations l
    where d.location_id = l.location_id
  ) dl
where e.department_id = dl.department_id

--------------------------------------------
--Multiple-Row Subqueries - IN
--------------------------------------------

SELECT * FROM employees
WHERE job_id IN
    (
        SELECT job_id FROM jobs
        WHERE 5000 between min_salary and max_salary
    )

SELECT customer_id, first_name, last_name
FROM customer
WHERE address_id IN
    (
        SELECT address_id FROM address WHERE city_id IN
        (
            SELECT city_id FROM city WHERE country_id = 50
        )
    )

SELECT film_id, title, rating
FROM film
WHERE
    film_id IN
    (
        SELECT i.film_id
        FROM inventory i
        INNER JOIN rental r ON i.inventory_id = r.inventory_id
        WHERE
            r.rental_date BETWEEN '2005-05-25' AND '2005-05-26'
    );
```

```sql
--------------------------------------------
--Multiple-Row Subqueries - IN (Multiple Columns)
--------------------------------------------

SELECT title, rental_duration, length
FROM film gf
WHERE rating = 'G'
  AND (gf.rental_duration, gf.length) IN
  (
    SELECT lf.rental_duration, lf.length
    FROM film lf
    WHERE length > 170
  )

--------------------------------------------
--Multiple-Row Subqueries - ANY
--------------------------------------------

SELECT
    first_name, last_name, salary
FROM employees
WHERE salary < ANY
    (
        SELECT salary
        FROM employees
        WHERE job_id = 9
    )

SELECT category_id, MAX(length)
FROM film
INNER JOIN film_category USING(film_id)
GROUP BY category_id
ORDER BY 2

SELECT title, length
FROM film
WHERE length > ANY
(
    SELECT MAX(length)
    FROM film
    INNER JOIN film_category USING(film_id)
    GROUP BY category_id
);

--------------------------------------------
--Multiple-Row Subqueries - ANY vs IN
--------------------------------------------

SELECT title, category_id
FROM film f, film_category c
WHERE
    1=1
    AND f.film_id = c.film_id
    AND category_id = ANY
    (
        SELECT category_id
        FROM category
        WHERE name in ('Animation', 'Games')
    );
```

```
--------------------------------------------
--Multiple-Row Subqueries - ALL
--------------------------------------------

SELECT first_name, last_name, salary
FROM employees
WHERE salary > ALL
    (
        SELECT salary
        FROM employees
        WHERE manager_id = 1
    )

SELECT film_id, title, length
FROM film
WHERE
    length <
    (
        SELECT MIN(avg_length) FROM
        (
            SELECT ROUND(AVG(length), 2) avg_length
            FROM film
            GROUP BY rating
        ) sb
    )
ORDER BY 3;

--------------------------------------------
--EXISTS Operator
--------------------------------------------

SELECT first_name, last_name
FROM customer c
WHERE
    EXISTS
    (
        SELECT 1
        FROM payment p
        WHERE p.customer_id = c.customer_id
    );

select * from departments d
where exists
  (
    select 1 from locations l
    where l.location_id = d.location_id
      and country_id in
        (
           select country_id from countries
           where country_name = 'Canada'
        )
  )
```

```
--------------------------------------------
--NOT EXISTS Operator
--------------------------------------------

SELECT * FROM employees e
WHERE NOT EXISTS
    (
        SELECT 1 FROM jobs j
        WHERE LOWER(job_title) LIKE '%manager%'
             AND j.job_id = e.job_id
    )
```

```
-----------------------------------------------------
--EXERCISE ANSWERS
-----------------------------------------------------

--------------------------------------------
--Single-Row Subqueries - Where
--------------------------------------------

SELECT city_id, city FROM city
WHERE country_id =
(
    SELECT country_id FROM country WHERE country = 'Japan'
);

--------------------------------------------
--Single-Row Subqueries - Column
--------------------------------------------

select first_name, last_name,
  (
    select first_name || ' ' || last_name as parent_name
    from employees e
    where e.employee_id = d.employee_id
  )
from dependents d

--------------------------------------------
--Single-Row Subqueries - Having
--------------------------------------------

select department_id,
      (select department_name from departments ds
      where ds.department_id = e.department_id) as dep_name,
      count(*) as emp_num
from employees e
group by department_id
having count(*) >
      (
            select count(*) from employees e
            where e. department_id = (select department_id
            from departments d where department_name = 'IT')
      )

--Alternative

select e.department_id, department_name, count(*) as emp_num
from employees e
inner join departments ds on ds.department_id = e.department_id
group by e.department_id, department_name
having count(*) >
      (
            select count(*) from employees e
            where e. department_id = (select department_id
            from departments d where department_name = 'IT')
      )
```

```
-------------------------------------------
--Multiple-Row Subqueries - IN
-------------------------------------------

--Exercise-1

SELECT film_id, title, rating
FROM film
WHERE
      film id IN
      (
            SELECT i.film_id
            FROM inventory i
            WHERE i. inventory_id IN
            (
                  SELECT r. inventory_id FROM rental r
                  WHERE
                        rental_date BETWEEN '2005-05-25' AND '2005-05-261'
            )
      );

--Exercise-2

SELECT * FROM employees e
WHERE e.employee_id in
(
      SELECT d.employee_id FROM dependents d
);

--Alternative

SELECT e.* FROM employees e
INNER JOIN dependents d ON e.employee_id = d.employee_id

-------------------------------------------
--Multiple-Row Subqueries - ANY
-------------------------------------------

SELECT
    first_name, last_name, salary
FROM employees
WHERE salary <
    (
        SELECT MAX(salary)
        FROM employees
        WHERE job_id = 9
    )

-------------------------------------------
--Multiple-Row Subqueries - ALL
-------------------------------------------
--Exercise-1

SELECT first_name, last_name, salary
FROM employees
WHERE salary >
    (
        SELECT MAX(salary)
        FROM employees
        WHERE manager_id = 100
    )
```

```
--Exercise-2

SELECT film_id, title, length
FROM film
WHERE
    length < ALL
    (
        SELECT ROUND(AVG(length), 2)
        FROM film
        GROUP BY rating
    )
ORDER BY 3 DESC;

--------------------------------------------
--EXISTS Operator
--------------------------------------------

--Exercise-1

SELECT first_name, last_name
FROM customer c
WHERE c.customer_id IN
    (
        SELECT p.customer_id
        FROM payment p
    );

--Exercise-2

SELECT * FROM employees
WHERE job_id IN
    (
        SELECT job_id FROM jobs
        WHERE LOWER(job_title) LIKE '%manager%'
    )

SELECT * FROM employees e
WHERE EXISTS
    (
        SELECT 1 FROM jobs j
        WHERE LOWER(job_title) LIKE '%manager%'
             AND j.job_id = e.job_id
    )
```

```sql
--------------------------------------------
--INSERT - Single Row
--------------------------------------------

INSERT INTO departments(department_id, department_name, location_id)
VALUES(13, 'Networking', 2400);

INSERT INTO basket_a(id_a, fruit_a)
VALUES(8, 'Plum');

INSERT INTO basket_a
VALUES(9, 'Watermelon');

--------------------------------------------
--INSERT - Single Row - Default Value
--------------------------------------------

INSERT INTO student(id, name, class, mark, gender, course_name)
VALUES(36, 'Adele', DEFAULT, DEFAULT, DEFAULT, 'Finance');

INSERT INTO student(id, name, class, mark, gender, course_name)
VALUES(37, 'Tarkan', DEFAULT, 78, 'male', 'Zoology');

INSERT INTO student(id, name, mark, course_name)
VALUES(38, 'Ava Max', 85, 'History');

--------------------------------------------
--INSERT - RETURNING Clause
--------------------------------------------

CREATE TABLE basket_c
(
    id_c SERIAL PRIMARY KEY,
    fruit_c VARCHAR(50) NOT NULL,
    calorie INTEGER DEFAULT 0
);

INSERT INTO basket_c(fruit_c)
VALUES('Orange')
RETURNING id_c;

INSERT INTO basket_c(fruit_c)
VALUES('Watermelon')
RETURNING id_c AS last_id;
```

```sql
--------------------------------------------
--INSERT Multiple Rows
--------------------------------------------

INSERT INTO departments(department_id, department_name, location_id)
VALUES
    (14, 'Academy', 1700),
    (15, 'Security', 1400),
    (16, 'Logistic', 2500),
    (17, 'Customer Experience', 2400);
INSERT INTO basket_c(fruit_c)
VALUES

    ('Lime'),
    ('Peach'),
    ('Quince'),
    ('Nectarine')
RETURNING *;

--------------------------------------------
--Copying Rows from Another Table
--------------------------------------------

INSERT INTO basket_c (fruit_c)
SELECT fruit_a
FROM basket_a
UNION
SELECT fruit_b
FROM basket_b

create table managers as
select * from employees
where 1=0

INSERT INTO managers
SELECT * FROM employees
WHERE job_id IN
    (
        SELECT job_id FROM jobs
        WHERE LOWER(job_title) LIKE '%manager%'
    )

--------------------------------------------
--UPDATE - Single Row
--------------------------------------------

UPDATE student
SET mark = 85
WHERE id = 3

UPDATE cars
SET brand = 'Mercedes Benz'
WHERE id = 1
```

```sql
--------------------------------------------
--UPDATE - Multiple Row
--------------------------------------------

UPDATE employees
SET salary = salary + 1000,
    manager_id = 22
WHERE employee_id IN (13,14);

UPDATE student
SET course_name = 'Finance'
WHERE course_name = 'Economics';

--------------------------------------------
--UPDATE - Returning
--------------------------------------------

UPDATE film
SET replacement_cost = 20,
   last_update = current_date
WHERE film_id = 100
RETURNING *

--------------------------------------------
--UPDATE - JOIN
--------------------------------------------

UPDATE product p
SET net_price = price - price * s.discount
FROM product_segment s
WHERE p.segment_id = s.id;

UPDATE employees e
SET salary = salary * 1.2
FROM jobs j
WHERE e.job_id = j.job_id
     AND LOWER(job_title) LIKE '%manager%'

--------------------------------------------
--DELETE - Single Row
--------------------------------------------

DELETE FROM basket_c
WHERE id_c = 5;

DELETE FROM staff
WHERE staff_id = 104;

DELETE FROM product
WHERE name = 'Oven';

DELETE FROM student
WHERE id = 36;

DELETE FROM student
WHERE name = 'Arnold';

DELETE FROM courses
WHERE id = 3;
```

```
--------------------------------------------
--DELETE - Multiple Row
--------------------------------------------

DELETE FROM basket_c
WHERE id_c IN (8, 9);

DELETE FROM basket_c
WHERE fruit_c = 'Orange';

DELETE FROM basket_c
WHERE fruit_c IN
   (SELECT fruit_b FROM basket_b)

DELETE FROM student
WHERE course_name = 'Finance';

--------------------------------------------
--DELETE - Returning
--------------------------------------------

DELETE FROM colors
RETURNING *;

DELETE FROM student
WHERE id = 17
RETURNING name, course_name;

DELETE FROM cars
WHERE price > 150000

--------------------------------------------
--DELETE - JOIN
--------------------------------------------

DELETE FROM movies m
USING movie_reviews mr
WHERE mr.movie_id = m.movie_id
```

```
-----------------------------------------------------
--EXERCISE ANSWERS
-----------------------------------------------------

--------------------------------------------
--INSERT - Single Row
--------------------------------------------

INSERT INTO staff(staff_id, first_name, last_name, hire_date, departure_date)
VALUES(300, 'Billie', 'Eilish', '2023-02-01', NULL);

INSERT INTO staff
VALUES(301, 'Alan', 'Walker', '2023-01-01', '2032-12-31');

--------------------------------------------
--UPDATE - JOIN
--------------------------------------------

UPDATE employees e
SET salary = salary * 1.2
FROM jobs j
WHERE e.job_id = j.job_id
     AND LOWER(job_title) LIKE '%manager%'

UPDATE employees
SET salary = salary * 1.2
WHERE job_id IN
    (
        SELECT job_id FROM jobs
        WHERE LOWER(job_title) LIKE '%manager%'
    )

--------------------------------------------
--DELETE - JOIN
--------------------------------------------

--Exercise-1

DELETE FROM movies m
USING movie_reviews mr
WHERE mr.movie_id = m.movie_id

DELETE FROM movies m
WHERE m.movie_id IN
(
    SELECT mr.movie_id FROM movie_reviews mr
);

DELETE FROM movies m
WHERE EXISTS
(
    SELECT 1 FROM movie_reviews mr
    WHERE m.movie_id = mr.movie_id
);

--Exercise-2

DELETE FROM product p
USING product_segment ps
WHERE p.segment_id = ps.id
    and ps.segment = 'Luxury'
```

```sql
--------------------------------------------
--Manage Transaction - COMMIT
--------------------------------------------

-- start a transaction
BEGIN;


UPDATE courses
SET published_date = published_date + interval '2 month'
WHERE course_id = 1;


INSERT INTO courses(course_name, description, published_date)
VALUES('Working with PostgreSQL PL/PGSQL', 'Database Programming', '2023-03-
10');


-- commit the change (or roll it back later)
COMMIT;

--------------------------------------------
--Manage Transaction - ROLLBACK
--------------------------------------------

-- start a transaction
BEGIN;


INSERT INTO movie_reviews(movie_id, review)
VALUES
    (10, 'Wonderful'),
    (8, 'Cool');

UPDATE movie_reviews
SET movie_id = 5
WHERE review_id = 6;


-- rollback the changes
ROLLBACK TRANSACTION;

--------------------------------------------

BEGIN;

UPDATE product_segment
SET discount = discount + 0.05;

SELECT * FROM product_segment;

UPDATE product p
SET net_price = price - price * s.discount
FROM product_segment s
WHERE p.segment_id = s.id;
```

```sql
SELECT SUM(net_price) FROM product;

ROLLBACK TRANSACTION;

---------------------------------------------
--Manage Transaction - SAVEPOINT
---------------------------------------------

-- start a transaction
BEGIN;

INSERT INTO movie_reviews(movie_id, review)
VALUES (10, 'Wonderful');

--New savepoint
SAVEPOINT dbhunter;

INSERT INTO movie_reviews(movie_id, review)
VALUES(8, 'Bad');

UPDATE movie_reviews
SET movie_id = 5 WHERE review_id = 6;

--Up to savepoint dbhunter changes are rollback
ROLLBACK TO SAVEPOINT dbHunter;
```

```sql
--------------------------------------------
--Creating a Table - Writing Scripts
--------------------------------------------

CREATE TABLE educations
(
    education_id SERIAL PRIMARY KEY,
    education_name VARCHAR(100) NOT NULL UNIQUE,
    description VARCHAR(500),
    start_date DATE,
    end_date DATE
)

--------------------------------------------

CREATE TABLE accounts
(
    user_id SERIAL PRIMARY KEY,
    username VARCHAR(70) UNIQUE NOT NULL,
    user_password VARCHAR(50) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    created_on TIMESTAMP NOT NULL DEFAULT CURRENT_DATE,
    last_login TIMESTAMP
);

--------------------------------------------
--Creating a Table - Select Into
--------------------------------------------

SELECT f.film_id, f.title, c.name
    INTO TABLE film_with_category
FROM film f, film_category fc, category c
WHERE
    1=1
    AND f.rating = 'R'
    AND fc.film_id = f.film_id
    AND fc.category_id = c.category_id
ORDER BY title;

--------------------------------------------

SELECT film_id, title, length
    INTO TEMP TABLE long_film
FROM film
WHERE length > 150
ORDER BY title;

--------------------------------------------
--Creating a Table - Create Table As (Query)
--------------------------------------------

CREATE TABLE horror_film AS
SELECT
    film_id, title,
    release_year,
    length, rating
FROM film
INNER JOIN film_category USING (film_id)
WHERE category_id = 11;
```

```
--------------------------------------------

CREATE TABLE IF NOT EXISTS
    film_ratings (rating, film_count) AS
SELECT rating, COUNT (film_id)
FROM film
GROUP BY rating;

--------------------------------------------
--Creating a Table - Create Table As (Table)
--------------------------------------------

CREATE TABLE movies_copy AS
TABLE movies;

--------------------------------------------

CREATE TABLE movies_copy2 AS
TABLE movies
WITH NO DATA;

--------------------------------------------

CREATE TABLE product_segment_new AS
TABLE product_segment
WITH NO DATA;

--------------------------------------------
--SERIAL pseudo-type
--------------------------------------------

CREATE TABLE colors2(
    colors_id SERIAL,
    color VARCHAR(30)
)

INSERT INTO colors2(colors_id, color)
VALUES(Default, 'Brown');

--------------------------------------------

pg_get_serial_sequence('table_name','column_name')

SELECT currval(pg_get_serial_sequence('colors2', 'colors_id'));

SELECT nextval(pg_get_serial_sequence('colors2', 'colors_id'));

--------------------------------------------
--SEQUENCE
--------------------------------------------

CREATE SEQUENCE sequence_temp
INCREMENT 10
START 100;

SELECT nextval('sequence_temp');
```

```sql
--------------------------------------------

CREATE SEQUENCE sequence_descending
INCREMENT -1
MINVALUE 1
MAXVALUE 100
START 10
CYCLE;

SELECT nextval('sequence_descending');

--------------------------------------------
--SEQUENCE - Associated with a table column-1
--------------------------------------------

CREATE TABLE payment_details(
    payment_id SERIAL,
    item_id INT NOT NULL,
    item_text VARCHAR(50) NOT NULL,
    price DEC(10,2) NOT NULL,
    PRIMARY KEY(payment_id, item_id)
);

CREATE SEQUENCE sq_payment_item_id
START 10
INCREMENT 10
MINVALUE 10
OWNED BY payment_details.item_id;

INSERT INTO
    payment_details(payment_id, item_id, item_text, price)
VALUES
    (100, nextval('sq_payment_item_id'),'Sofa',120),
    (100, nextval('sq_payment_item_id'),'Fridge',550),
    (100, nextval('sq_payment_item_id'),'Speaker',50),
    (101, nextval('sq_payment_item_id'),'Table',250),
    (101, nextval('sq_payment_item_id'),'Lamp',25);

--------------------------------------------
--SEQUENCE - Associated with a table column-2
--------------------------------------------

CREATE SEQUENCE sq_payment_item_id
START 10
INCREMENT 10
MINVALUE 10

CREATE TABLE payment_details(
    payment_id SERIAL,
    item_id INT NOT NULL DEFAULT nextval('sq_payment_item_id'),
    item_text VARCHAR(50) NOT NULL,
    price DEC(10,2) NOT NULL,
    PRIMARY KEY(payment_id, item_id)
);


ALTER SEQUENCE sq_payment_item_id
OWNED BY payment_details.item_id;
```

```sql
INSERT INTO
    payment_details(payment_id, item_text, price)
VALUES
    (100,'Sofa',120),
    (100, 'Fridge',550),
    (100, 'Speaker',50),
    (101, 'Table',250),
    (101, 'Lamp',25);
```

```sql
--------------------------------------------
--ADD Columns with Script
--------------------------------------------

ALTER TABLE staff
ADD COLUMN email varchar(100);

--------------------------------------------

ALTER TABLE staff
ADD COLUMN salary numeric(8,2) DEFAULT 0;

--------------------------------------------

ALTER TABLE movie_reviews
ADD COLUMN review_date date,
ADD COLUMN last_update date DEFAULT CURRENT_DATE,
ADD COLUMN review_rate int DEFAULT 1;

--------------------------------------------
--RENAME a Column
--------------------------------------------

ALTER TABLE movie_reviews
RENAME COLUMN review_date to user_review_date;

--------------------------------------------

ALTER TABLE student
RENAME COLUMN class to class_no;

--------------------------------------------
--Change Default Value
--------------------------------------------

ALTER TABLE movie_reviews
ALTER COLUMN review
SET DEFAULT 'Good';

--------------------------------------------

ALTER TABLE movie_reviews
ALTER COLUMN review_rate
DROP DEFAULT;

--------------------------------------------
--Change Column Type
--------------------------------------------

ALTER TABLE courses
ALTER COLUMN description TYPE TEXT;

--------------------------------------------

ALTER TABLE colors
ALTER COLUMN color TYPE VARCHAR(20);

--------------------------------------------

ALTER TABLE budgets
ALTER COLUMN current_year TYPE smallint,
ALTER COLUMN previous_year TYPE smallint;
```

```sql
--------------------------------------------
--ADD Comment
--------------------------------------------

comment on table cars is 'This table contains car information';
comment on table educations is 'All education information is here';

--------------------------------------------

comment on column cars.brand is 'Car brand';
comment on column product.net_price is 'In this column, net price information
is kept';

--------------------------------------------
--Comment List
--------------------------------------------

select * from
(
  select t.table_schema, t.table_name,
    pg_catalog.obj_description(pgc.oid, 'pg_class') as table_comment
  from information_schema.tables t
    inner join pg_catalog.pg_class pgc
      on t.table_name = pgc.relname
  where t.table_type='BASE TABLE'
    and t.table_schema='public'
) sub
where table_comment is not null;

--------------------------------------------

select
  c.table_schema, c.table_name, c.column_name,
  pgd.description as column_comment
from pg_catalog.pg_statio_all_tables as st
  inner join pg_catalog.pg_description pgd
    on pgd.objoid = st.relid
  inner join information_schema.columns c
    on pgd.objsubid   = c.ordinal_position
      and c.table_schema = st.schemaname
      and c.table_name   = st.relname;

--------------------------------------------
--Add - Drop NOT NULL
--------------------------------------------

ALTER TABLE movie_reviews
ALTER COLUMN review_rate
SET NOT NULL;

--------------------------------------------

ALTER TABLE movie_reviews
ALTER COLUMN review_rate
DROP NOT NULL;
```

```
--------------------------------------------
--Drop Columns
--------------------------------------------

ALTER TABLE movie_reviews
DROP COLUMN review_date,
DROP COLUMN last_update;

--------------------------------------------
--CHECK Constraint
--------------------------------------------

ALTER TABLE movie_reviews
ADD CHECK (review_rate between 1 and 5)

--------------------------------------------
--ADD Constraint
--------------------------------------------

ALTER TABLE courses
ADD CONSTRAINT uq_courses UNIQUE (course_name);

--------------------------------------------
--DROP Constraint
--------------------------------------------

ALTER TABLE courses
DROP CONSTRAINT uq_courses;

--------------------------------------------
--Rename Table
--------------------------------------------

ALTER TABLE courses
RENAME TO courses_technical;

--------------------------------------------
--DROP Table
--------------------------------------------

DROP TABLE IF EXISTS budgets2;

--------------------------------------------

DROP TABLE authors, pages;

--------------------------------------------

CREATE TABLE authors (
    author_id SMALLSERIAL PRIMARY KEY,
    firstname VARCHAR (50),
    lastname VARCHAR (50)
);

CREATE TABLE pages (
    page_id SMALLSERIAL PRIMARY KEY,
    title VARCHAR (255) NOT NULL,
    author_id INT NOT NULL,
    FOREIGN KEY (author_id)
        REFERENCES authors (author_id)
);
```

```sql
DROP TABLE IF EXISTS authors;

DROP TABLE authors CASCADE;

--------------------------------------------
--TRUNCATE Table
--------------------------------------------

TRUNCATE TABLE colors;
```

```
-----------------------------------------------------
--EXERCISE ANSWERS
-----------------------------------------------------

--------------------------------------------
--Alter Tables
--------------------------------------------

-- Add primary key for artist_id column
alter table artist add constraint atrist_pk primary key(artist_id);

-- Fix column names
alter table artist rename column firstname to first_name
alter table artist rename column lastname to last_name

-- Add date of birth column
alter table artist add column date_of_birth date

-- Add address column (max 200 chars)
alter table artist add column address varchar(200)

-- Add city column with not null constraint
alter table artist add column city varchar(100) not null

-- Add zip code column (max value is 9999)
alter table artist add zip_code int check (zip_code < 9999)

-- Add company column (default value is Warner Bros)
alter table artist add column company varchar(100) default 'Warner Bros'

-- Address column change type to varchar(500)
alter table artist alter column address type varchar(500)

-- Add not null constraint to zip code column
alter table artist alter column zip_code set not null

-- Change table name: artist_info
alter table artist rename to artist_info
```

```sql
--------------------------------------------
--Primary Key Constraint
--------------------------------------------

CREATE TABLE authors (
    author_id INT PRIMARY KEY,
    firstname VARCHAR (50),
    lastname VARCHAR (50)
);

CREATE TABLE products (
    product_no INTEGER,
    description TEXT,
    product_cost NUMERIC
);

ALTER TABLE products
ADD PRIMARY KEY (product_no);

INSERT INTO authors
VALUES (1, 'Stephen', 'King'), (2, 'Agatha', 'Christie');

INSERT INTO authors
VALUES (2, 'Lev', 'Tolstoy');

--------------------------------------------

CREATE TABLE car_brands (
    brand_name VARCHAR(50)
);

INSERT INTO car_brands (brand_name)
VALUES
    ('Peugeot'),
    ('Mercedes'),
    ('Ford'),
    ('FIAT');

ALTER TABLE car_brands
ADD COLUMN car_brands_id SERIAL PRIMARY KEY;

--------------------------------------------
--Foreign Key Constraint
--------------------------------------------

insert into dependents (dependent_id, first_name, last_name, relationship,
employee_id)
values (32,'Melisa','Grant','Child', 100);

update dependents set employee_id = 21 where employee_id = 100

--DROP
ALTER TABLE dependents DROP CONSTRAINT dependents_employee_id_fey

--ADD
ALTER TABLE dependents
ADD CONSTRAINT dependents_employee_id_fkey
FOREIGN KEY (employee_id)
REFERENCES employees (employee_id);
```

```sql
--------------------------------------------

CREATE TABLE account_roles
(
    user_id INT NOT NULL,
    role_id INT NOT NULL,
    grant_date TIMESTAMP,
    PRIMARY KEY (user_id, role_id),
    FOREIGN KEY (role_id)
      REFERENCES roles (role_id),
    FOREIGN KEY (user_id)
      REFERENCES accounts (user_id)
);

CREATE TABLE roles
(
    role_id serial PRIMARY KEY,
    role_name VARCHAR (255) UNIQUE NOT NULL
);

CREATE TABLE accounts
(
    user_id SERIAL PRIMARY KEY,
    username VARCHAR(70) UNIQUE NOT NULL,
    user_password VARCHAR(50) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    created_on TIMESTAMP NOT NULL DEFAULT CURRENT_DATE,
    last_login TIMESTAMP
);

--------------------------------------------

CREATE TABLE customers(
    customer_id SERIAL,
    customer_name VARCHAR(255) NOT NULL,
    PRIMARY KEY(customer_id)
);

CREATE TABLE contacts(
    contact_id SERIAL,
    customer_id INT,
    contact_name VARCHAR(255) NOT NULL,
    PRIMARY KEY(contact_id),
    CONSTRAINT fk_customer
        FOREIGN KEY(customer_id)
        REFERENCES customers(customer_id)
);

INSERT INTO customers (customer_name)
VALUES('Apple'), ('Google'), ('Amazon'), ('IBM');

INSERT INTO contacts (customer_id, contact_name)
VALUES(1, 'Alex'), (2, 'Michael'), (2, 'Jane'), (3, 'Judi');
```

```
--------------------------------------------
--Foreign Key Constraint - ON DELETE
--------------------------------------------

DELETE FROM customers WHERE customer_id = 1;

--------------------------------------------
--Foreign Key Constraint - ADD-DROP
--------------------------------------------

CREATE TABLE contacts(
    contact_id SERIAL PRIMARY KEY,
    customer_id INT,
    contact_name VARCHAR(255) NOT NULL,
     CONSTRAINT fk_customer
        FOREIGN KEY(customer_id)
        REFERENCES customers(customer_id)
);

ALTER TABLE child_table
ADD CONSTRAINT constraint_name
      FOREIGN KEY (fk_columns)
      REFERENCES parent_table (parent_key_columns);

ALTER TABLE child_table
DROP CONSTRAINT constraint_fkey;

--------------------------------------------
--CHECK Constraint
--------------------------------------------

CREATE TABLE employees_new (
    id SERIAL PRIMARY KEY,
    first_name VARCHAR (50),
    last_name VARCHAR (50),
    birth_date DATE CHECK (birth_date > '1980-01-01'),
    joined_date DATE CHECK (joined_date > birth_date),
    salary NUMERIC CHECK(salary between 5000 and 10000),
    emp_type VARCHAR(1) CHECK(emp_type in ('A', 'B', 'C'))
);

INSERT INTO employees_new (first_name, last_name, birth_date, joined_date, salary,
emp_type)
VALUES('David', 'Austin', '2001-01-12', '2021-09-22', 8000, 'C');

--------------------------------------------

ALTER TABLE employees
ADD CONSTRAINT emp_type_check
CHECK(emp_type in ('A', 'B', 'C'));

ALTER TABLE film
ADD CONSTRAINT rental_dur_check
CHECK(rental_duration < 10);
```

```sql
--------------------------------------------
--UNIQUE Constraint
--------------------------------------------

CREATE TABLE person (
    id SERIAL PRIMARY KEY,
    first_name VARCHAR (50),
    last_name VARCHAR (50),
    email VARCHAR (50) UNIQUE
);

CREATE TABLE person (
    id SERIAL  PRIMARY KEY,
    first_name VARCHAR (50),
    last_name  VARCHAR (50),
    email      VARCHAR (50),
    UNIQUE(email)
);

ALTER TABLE person
ADD CONSTRAINT uq_email UNIQUE(email)

--------------------------------------------

CREATE TABLE customer_new
(
    customer_id serial NOT NULL,
    first_name character varying(45),
    last_name character varying(45),
    email character varying(50),
    address_id smallint NOT NULL,
    UNIQUE(first_name, last_name, address_id)
)

insert into customer_new(first_name, last_name, address_id)
values('Johnny', 'Depp', 1234), ('Kevin', 'Spacey', 4567);

--------------------------------------------
--NOT NULL Constraint
--------------------------------------------

CREATE TABLE person (
    id SERIAL PRIMARY KEY,
    first_name VARCHAR (50) NOT NULL,
    last_name VARCHAR (50) NOT NULL,
    email VARCHAR (50) UNIQUE
);

ALTER TABLE customer
ALTER COLUMN first_name SET NOT NULL,
ALTER COLUMN last_name SET NOT NULL;

--------------------------------------------

INSERT INTO person (first_name, last_name, email)
VALUES
    ('Jennifer', 'Davis', NULL),
    ('Joe', 'Swank', 'joe.swank@dbhunter.com');
```

```sql
ALTER TABLE person
ALTER COLUMN email SET NOT NULL;

--------------------------------------------

CREATE TABLE users
(
    id serial PRIMARY KEY,
    username VARCHAR (50),
    email VARCHAR (50),
    CONSTRAINT username_email_not_null
    CHECK
    (
        NOT
        (
            ( username IS NULL  OR  username = '' )
            AND
            ( email IS NULL  OR  email = '' )
        )
    )
);

INSERT INTO users (username, email)
VALUES
    ('user1', NULL),
    (NULL, 'email1@example.com'),
    ('user2', 'email2@example.com'),
    ('user3', '');

INSERT INTO users (username, email)
VALUES
    (NULL, NULL),
    (NULL, ''),
    ('', NULL),
    ('', '');
```

```
-----------------------------------------------------
--EXERCISE ANSWERS
-----------------------------------------------------

--------------------------------------------
--Constraints
--------------------------------------------

drop table category_books;
drop table books;
drop table category;

-- Books Table:
-- book id - primary key
-- title
-- price - value between 10 and 1000
-- author_id - not null

create table books
(
      book_id smallserial primary key,
      title varchar(200),
      price numeric(6,2) check(price between 10 and 1000),
      author_id int not null
)

alter table books
add constraint fk_author_id
      foreign key (author_id)
      references authors(author_id);

-- Authors Table:
-- authorId - primary key
-- first name - not null
-- last name - not null
-- email - unique
create table authors
(
      author_id smallserial primary key,
      first_name varchar(50) not null,
      last_name varchar(50) not null,
      email varchar(150) unique
)

-- Category Table:
-- category id - primary key
-- category name - not null
create table category
(
      category_id smallserial primary key,
      category_name varchar(50) not null
)
```

```sql
-- Category_Books Table
-- category id - foreign key references category table
-- book id - foreign key references books table
create table category_books
(
      category_id int,
      book_id int,
      constraint fk_category_id
            foreign key(category_id)
            references category(category_id),
      constraint fk_book_id
            foreign key(book_id)
            references books(book_id)
);

insert into authors (first_name,last_name, email) values
       ('Stephen','King', 'stephen.king@abc.com'),
       ('Agatha','Christie', 'agatha.christie@abc.com'),
       ('Leo','Tolstoy', 'leo.tolstoy@abc.com');

insert into books (title, price, author_id) values
       ('The Shining', 11, 1), ('Nightshift', 15, 1), ('The Dead Zone', 30, 1),
       ('The Secret Adversary', 34, 2), ('The Secret of Chimneys', 45, 2), ('The
Mystery of the Blue Train', 60, 2),
       ('Anna Karenina', 78, 3), ('War and Peace', 120, 3), ('What Is Art', 12, 3);

insert into category(category_name) values
      ('Adventure stories'), ('Classics'), ('Crime'), ('Fantasy'), ('Horror'),
      ('Mystery'), ('Romance'), ('Science fiction'), ('Short stories'), ('Plays')

insert into category_books values
      (1,1), (1,2),(3,2),(4,3),(4,4),(6,6),(6,7),(8,2),(9,9)
```

```sql
--------------------------------------------
--Creating a View
--------------------------------------------

CREATE VIEW vw_emp_info AS
SELECT
    e.employee_id, e.first_name, e.last_name,
    j.job_title, d.department_name
FROM employees e
    INNER JOIN jobs j USING(job_id)
    INNER JOIN departments d USING(department_id);

--------------------------------------------

CREATE VIEW vw_cust_info AS
SELECT cu.customer_id,
    cu.first_name || ' ' || cu.last_name AS full_name,
    co.country, ci.city,ad.address,
    ad.postal_code, ad.phone, cu.store_id,
    CASE
        WHEN cu.activebool THEN 'OK'
    ELSE 'NOK'
    END AS active,
    (SELECT COUNT(*) FROM rental re
     WHERE re.customer_id = cu.customer_id) rental_number
FROM customer cu
    INNER JOIN address ad USING (address_id)
    INNER JOIN city ci USING (city_id)
    INNER JOIN country co USING (country_id);

--------------------------------------------

CREATE VIEW vw_emp_info AS
SELECT
    e.employee_id, e.first_name, e.last_name,
    j.job_title, d.department_name
FROM employees e
    INNER JOIN jobs j USING(job_id)
    INNER JOIN departments d USING(department_id);

--------------------------------------------
--DROP a View
--------------------------------------------

DROP VIEW IF EXISTS vw_emp_info;

DROP VIEW IF EXISTS vw_cust_info;

--------------------------------------------
--DROP a View - Dependency
--------------------------------------------

CREATE VIEW main_film AS
SELECT film_id, title,
    length, c.name category
FROM film
    INNER JOIN film_category USING (film_id)
    INNER JOIN category c USING(category_id);
```

```sql
CREATE VIEW horror_film AS
SELECT film_id, title, length
FROM main_film
WHERE category = 'Horror';

DROP VIEW main_film;

DROP VIEW main_film CASCADE;

--------------------------------------------
--Updatable Views
--------------------------------------------

create view student_engineer as
select * from student
where course_name ilike '%engineer%'

update student_engineer
set email = regexp_replace(lower(name), '[\s+]', '', 'g') || '@engineer.com'

--------------------------------------------

create view vw_product_grand_lux as
select id, name, price from product
where segment_id = 1

update vw_product_grand_lux
set price = price * 1.3

delete from vw_product_grand_lux
where id = 11;

insert into vw_product_grand_lux(id, name, price)
values(21, 'Air Cleaner', 1200)

--------------------------------------------
--Views - WITH CHECK OPTION
--------------------------------------------

CREATE VIEW vw_usa_city AS
SELECT country_id, city_id, city
FROM city
WHERE country_id = 103
ORDER BY city;

INSERT INTO vw_usa_city (country_id, city, city_id)
VALUES (102,'Birmingham', 601);

UPDATE vw_usa_city
SET country_id = 104 WHERE city_id = 11

--------------------------------------------

CREATE VIEW vw_usa_city AS
SELECT country_id, city_id, city
FROM city
WHERE country_id = 103
ORDER BY city
WITH CHECK OPTION;
```

```sql
INSERT INTO vw_usa_city (country_id, city_id, city)
VALUES (102, 602, 'Birmingham');

UPDATE vw_usa_city
SET country_id = 104 WHERE city_id = 33

--------------------------------------------
--Views - WITH CASCADED CHECK OPTION
--------------------------------------------

CREATE VIEW vw_city_b AS
SELECT city_id, city, country_id
FROM city
WHERE city LIKE 'B%';

CREATE OR REPLACE VIEW vw_city_b_usa AS
SELECT city_id, city, country_id
FROM vw_city_b
WHERE country_id = 103
WITH CASCADED CHECK OPTION;

INSERT INTO vw_city_b_usa (city_id, city, country_id)
VALUES (620, 'Dallas', 103);

--------------------------------------------
--Views - WITH LOCAL CHECK OPTION
--------------------------------------------

CREATE OR REPLACE VIEW vw_city_b_usa AS
SELECT city_id, city, country_id
FROM vw_city_b
WHERE country_id = 103
WITH LOCAL CHECK OPTION;

INSERT INTO vw_city_b_usa (city_id, city, country_id)
VALUES (620, 'Houston', 103);
```

```
-----------------------------------------------------
--EXERCISE ANSWERS
-----------------------------------------------------

--------------------------------------------
--Creating a View
--------------------------------------------

CREATE OR REPLACE VIEW vw_emp_info2 AS
SELECT
    e.first_name ||' '|| e.last_name as man_name,
    m.first_name ||' '|| m.last_name as emp_name,
    (
      select count(*) from dependents d
      where d.employee_id = e.employee_id
    ) dep_count
FROM employees m, employees e
WHERE e.employee_id = m.manager_id
```

```sql
--------------------------------------------
--CREATE Index
--------------------------------------------

EXPLAIN
SELECT * FROM address
WHERE phone = '223664661973';

CREATE INDEX idx_address_phone
ON address(phone);

--------------------------------------------


EXPLAIN
SELECT * FROM film
WHERE description = 'drama';

CREATE INDEX idx_film_desc
ON film(description);

--------------------------------------------
--CREATE Index - Multiple Column
--------------------------------------------

EXPLAIN
SELECT * FROM staff_test
WHERE first_name = 'Luis' AND last_name = 'Popp'

CREATE INDEX idx_staff_test_names
ON staff_test (first_name, last_name);

EXPLAIN
SELECT * FROM staff_test
WHERE first_name = 'Luis';

--------------------------------------------
--UNIQUE Indexes
--------------------------------------------

CREATE UNIQUE INDEX idx_employees_email ON employees(email);

INSERT INTO employees(first_name, last_name, email, hire_date, job_id,
salary)
VALUES ('Daniel', 'Faviet Urman', 'daniel.faviet@sqltutorial.org',
current_date, 4, 5000)

--------------------------------------------


CREATE UNIQUE INDEX idx_cust_name
ON customer(first_name, last_name, email);

CREATE UNIQUE INDEX idx_payment
ON payment(customer_id, rental_id);
```

```sql
--------------------------------------------
--Index on Expression
--------------------------------------------

EXPLAIN
SELECT
    customer_id, store_id
    first_name, last_name
FROM customer
WHERE last_name = 'Ely';

EXPLAIN
SELECT
    customer_id, store_id
    first_name, last_name
FROM customer
WHERE lower(last_name) = 'ely';

CREATE INDEX idx_ic_last_name
ON customer(LOWER(last_name));

--------------------------------------------
--Partial Index
--------------------------------------------

EXPLAIN
SELECT
    customer_id,
    first_name, last_name
FROM customer
WHERE active = 0;

CREATE INDEX idx_customer_inactive
ON customer(active)
WHERE active = 0;

--------------------------------------------
--DROP Index
--------------------------------------------

DROP INDEX idx_emp_job_id

--------------------------------------------
--List Indexes
--------------------------------------------

--List of all indexes
SELECT tablename, indexname, indexdef
FROM pg_indexes
WHERE schemaname = 'public'
ORDER BY 1, 2;

--Usage statistics of indexes

SELECT schemaname, relname,
       indexrelname, idx_scan
FROM pg_stat_user_indexes
ORDER BY idx_scan;
```

```
-----------------------------------------------------
--EXERCISE ANSWERS
-----------------------------------------------------

--------------------------------------------
--Create Index
--------------------------------------------

Table: address
Column(s): district
Type: Normal

CREATE INDEX idx_address_district ON address(district);

-------------------------------------------------------------

Table: actor
Column(s): first_name, last_name
Type: Normal

CREATE INDEX idx_actor_first_last_name ON actor(first_name, last_name);

-------------------------------------------------------------

Table: student
Column(s): name, course_name
Type: Unique

CREATE UNIQUE INDEX idx_student_name_cname ON student(name, course_name);

-------------------------------------------------------------

Table: product
Column(s): segment_id
Type: Partial (segment_id = 1)

CREATE INDEX idx_part_product_segment_id ON product(segment_id)
WHERE segment_id = 1
```