Views

62. Look at the `yum` table:

```
SELECT * FROM yum;
```

---

63. Query the `yum` table, aggregating by year and month:

```
SELECT
    EXTRACT(YEAR FROM date)::INT AS year,
    EXTRACT(MONTH FROM date)::INT AS month,
    AVG(open) AS avg_open,
    AVG(high) AS avg_high,
    AVG(low) AS avg_low,
    AVG(close) AS avg_close,
    SUM(volume) AS total_volume
FROM yum
GROUP BY 1, 2
ORDER BY 1, 2;
```

---

64. Save the results as a view named `yum_by_month`:

```
CREATE VIEW yum_by_month AS
SELECT
    EXTRACT(YEAR FROM date)::INT AS year,
    EXTRACT(MONTH FROM date)::INT AS month,
    AVG(open) AS avg_open,
    AVG(high) AS avg_high,
    AVG(low) AS avg_low,
    AVG(close) AS avg_close,
    SUM(volume) AS total_volume
FROM yum
GROUP BY 1, 2
ORDER BY 1, 2;
```

---

65. Create a view named `trans_by_month`:

```
CREATE VIEW trans_by_month AS
SELECT
    EXTRACT(YEAR FROM orderdate)::INT AS year,
    EXTRACT(MONTH FROM orderdate)::INT AS month,
    SUM(unit_price * quantity) AS total_sales
FROM transactions
GROUP BY 1, 2
ORDER BY 1, 2;
```

---

66. Create a view named `trans_by_employee`:

```
CREATE VIEW trans_by_employee AS
SELECT
    employee_id,
    SUM(unit_price * quantity) AS total_sales
```

```
FROM transactions
GROUP BY employee_id;
```

CTEs

67. Most common first initial for pets:

```
WITH initials AS (
    SELECT LOWER(SUBSTR(name, 1, 1)) AS initial
    FROM pets
)
SELECT
    initial,
    COUNT(*) AS count
FROM initials
GROUP BY initial
ORDER BY count DESC
LIMIT 1;
```

---

68. Create taglines for employees:

```
WITH employee_data AS (
    SELECT
        firstname || ' ' || lastname AS name,
        CASE
            WHEN job = 'IT' THEN job
            ELSE LOWER(job)
        END AS formatted_job,
        TO_CHAR(salary, 'FM$999,999,999') AS formatted_salary,
        EXTRACT(YEAR FROM startdate)::INT AS year_started
    FROM employees
)
SELECT
    name || ' started in ' || year_started || ' and makes ' || formatted_salary || ' working in ' ||
formatted_job || '.' AS tagline
FROM employee_data;
```

---

69. Revenue by company type (`LLC` , `Inc` , etc.):

```
WITH company_type_revenue AS (
    SELECT
        CASE
            WHEN customer LIKE '%LLC' THEN 'LLC'
            WHEN customer LIKE '%Inc' THEN 'Inc'
            WHEN customer LIKE '%Ltd' THEN 'Ltd'
            WHEN customer LIKE '%PLC' THEN 'PLC'
            ELSE 'Other'
        END AS company_type,
        SUM(unit_price * quantity) AS revenue
    FROM transactions
    GROUP BY company_type
)
SELECT
    company_type,
    COUNT(*) AS transaction_count,
    SUM(revenue) AS total_revenue
FROM company_type_revenue
GROUP BY company_type;
```

Joins

70. Which employee made which sale:

```
SELECT
    e.firstname,
    e.lastname,
    t.*
FROM transactions t
JOIN employees e ON t.employee_id = e.employee_id;
```

71. Employee who made the most in sales:

```
SELECT
    e.firstname || ' ' || e.lastname AS employee_name,
    SUM(t.unit_price * t.quantity) AS total_sales
FROM transactions t
JOIN employees e ON t.employee_id = e.employee_id
GROUP BY e.employee_id
ORDER BY total_sales DESC
LIMIT 1;
```

72. Solve using `trans_by_employee`:

```
SELECT
    e.firstname || ' ' || e.lastname AS employee_name,
    te.total_sales
FROM trans_by_employee te
JOIN employees e ON te.employee_id = e.employee_id
ORDER BY te.total_sales DESC
LIMIT 1;
```

73. Solve using a CTE:

```
WITH sales_by_employee AS (
    SELECT
        employee_id,
        SUM(unit_price * quantity) AS total_sales
    FROM transactions
    GROUP BY employee_id
)
SELECT
    e.firstname || ' ' || e.lastname AS employee_name,
    sbe.total_sales
FROM sales_by_employee sbe
JOIN employees e ON sbe.employee_id = e.employee_id
ORDER BY sbe.total_sales DESC
LIMIT 1;
```

74. Employees earning more from sales than 1.5 times their salary:

```
WITH sales_by_employee AS (
    SELECT
```

```
        employee_id,
        SUM(unit_price * quantity) AS total_sales
    FROM transactions
    GROUP BY employee_id
)
SELECT
    e.firstname || ' ' || e.lastname AS employee_name,
    e.salary,
    sbe.total_sales
FROM sales_by_employee sbe
JOIN employees e ON sbe.employee_id = e.employee_id
WHERE sbe.total_sales > 1.5 * e.salary;
```

---

75. Transactions before employee was hired:

```
SELECT
    t.order_id,
    t.orderdate,
    e.firstname || ' ' || e.lastname AS employee_name,
    e.startdate
FROM transactions t
JOIN employees e ON t.employee_id = e.employee_id
WHERE t.orderdate < e.startdate;
```

---

76. Monthly revenue vs Yum! trade volume:

```
SELECT
    EXTRACT(YEAR FROM orderdate)::INT AS year,
    EXTRACT(MONTH FROM orderdate)::INT AS month,
    SUM(t.unit_price * t.quantity) AS company_revenue,
    SUM(y.volume) AS yum_trade_volume
FROM transactions t
LEFT JOIN yum y
    ON EXTRACT(YEAR FROM t.orderdate) = EXTRACT(YEAR FROM y.date)
    AND EXTRACT(MONTH FROM t.orderdate) = EXTRACT(MONTH FROM y.date)
GROUP BY 1, 2
ORDER BY 1, 2;
```

---

77. Include lowest and highest price for Yum! stock:

```
SELECT
    EXTRACT(YEAR FROM orderdate)::INT AS year,
    EXTRACT(MONTH FROM orderdate)::INT AS month,
    SUM(t.unit_price * t.quantity) AS company_revenue,
    SUM(y.volume) AS yum_trade_volume,
    MIN(y.low) AS yum_lowest_price,
    MAX(y.high) AS yum_highest_price
FROM transactions t
LEFT JOIN yum y
    ON EXTRACT(YEAR FROM t.orderdate) = EXTRACT(YEAR FROM y.date)
    AND EXTRACT(MONTH FROM t.orderdate) = EXTRACT(MONTH FROM y.date)
GROUP BY 1, 2
ORDER BY 1, 2;
```
