1.Detecting Potential Payment fraud in an online Marketplace

```
SELECT
  user_id,
  COUNT(*) AS failed_transactions,
  COUNT(DISTINCT payment_method) AS distinct_payment_methods
FROM transactions
WHERE status = 'Failed'
GROUP BY user_id
HAVING COUNT(*) > 5 ;
```

2.average response time

```
SELECT
  ROUND(AVG(TIMESTAMP DIFF(HOUR, created_at, resolved_at)), 2) AS average_response_time
FROM support_tickets
WHERE resolved_at IS NOT NULL ;
```

3/////////. Query: Highest-Spending Customers per City

```
SELECT c.id AS customer_id,
       c.name,
       c.city,
       FLOOR(SUM(o.amount)) AS total_spending
FROM customers c
JOIN orders o ON c.id = o.customer_id
GROUP BY c.id, c.name, c.city
HAVING FLOOR(SUM(o.amount)) = (
    SELECT MAX(city_total)
    FROM (
        SELECT c2.city, FLOOR(SUM(o2.amount)) AS city_total
        FROM customers c2
        JOIN orders o2 ON c2.id = o2.customer_id
        WHERE c2.city = c.city
        GROUP BY c2.id, c2.name, c2.city
    ) AS sub
);
```

-- //4.Ecommerence product request report

```
SELECT
    p.name AS product_name,
    COUNT(r.product_id) AS total_requests
FROM products p
LEFT JOIN requests r ON p.id = r.product_id
WHERE p.is_available = 1
GROUP BY p.id, p.name
ORDER BY total_requests DESC, product_name ASC;
```

--5.active compaign engagement report

```
SELECT

    c.name,

    COUNT(e.id) AS total_engagements,

    SUM(e.views) AS total_views,

    SUM(e.clicks) AS total_clicks

FROM campaigns c

JOIN engagements e ON c.id = e.campaign_id

WHERE c.is_active = 1

GROUP BY c.id, c.name;
```

--6.tax report summary

```
    SELECT
    a.email,
    round(SUM(r.amount),2) AS total_report_amount
FROM accounts a
inner JOIN reports r ON r.account_id = a.id
```

```
WHERE year(r.dt) = '2023'   -- keep only 2023
GROUP BY a.email
ORDER BY a.email;
```

-- 7.Final Query (Antivirus Device Scan Report)

```
SELECT
    d.mac_address,
    COUNT(*) AS total_files_scanned,
    SUM(CASE WHEN sf.is_infected = 1 THEN 1 ELSE 0 END) AS total_infected_files
FROM devices d
JOIN scanned_files sf
    ON d.id = sf.device_id
GROUP BY d.mac_address
ORDER BY d.mac_address asc;
```

-- 8.Cryptocurrency Transactions Report

```
SELECT
    c.name AS coin_name,
    ROUND(SUM(t.amount), 2) AS total_transaction_amount,
    COUNT(*) AS total_transactions
FROM coins c
JOIN transactions t
    ON c.id = t.coin_id
WHERE YEAR(STR_TO_DATE(t.dt, '%Y-%m-%d %H:%i:%s')) = 2023
GROUP BY c.name
ORDER BY c.name;
```

-- 9.Customer Domain Ownership Report

```
SELECT
    c.email,
    COUNT(d.name) AS total_domains
```

```
FROM customers c
JOIN domains d
   ON c.id = d.customer_id
GROUP BY c.email
ORDER BY c.email;
```

--10.E-commerce wishlist report

```
SELECT

   p.name,

   FORMAT(p.price, 2) AS price,

   COUNT(w.customer_email) AS count

FROM products p

LEFT JOIN wishlists w ON p.id = w.product_id

WHERE p.in_stock = 1

GROUP BY p.id, p.name, p.price

ORDER BY p.name ASC;
```

- 11 – Email campaign report

```
 SELECT
    c.name AS campaign_name,
    SUM(e.emails_sent) AS total_emails_sent,
    SUM(e.emails_opened) AS total_emails_opened,
    (SUM(e.emails_sent) - SUM(e.emails_opened)) AS total_emails_not_opened
 FROM
    campaigns c
 JOIN
    email_stats e ON c.id = e.campaign_id
 GROUP BY
    c.name
```

```
ORDER BY
    c.name;
```

--12—Auction lot offers report

```
SELECT
    l.name AS lot_name,
    CAST(MAX(o.amount) AS DECIMAL(10,2)) AS highest_offer,
    COUNT(o.amount) AS total_offers
FROM
    lots l
JOIN
    offers o ON l.id = o.lot_id
GROUP BY
    l.name
ORDER BY
    l.name;
```

--13—Online Banking Transactions report

```
SELECT
    a.iban,
    FORMAT(MIN(t.amount), 'N2') AS min_transaction,
    FORMAT(MAX(t.amount), 'N2') AS max_transaction,
    FORMAT(AVG(t.amount), 'N2') AS avg_transaction,
    COUNT(*) AS total_transactions
FROM accounts a
JOIN transactions t
    ON a.id = t.account_id
WHERE t.dt >= '2022-09-01'
  AND t.dt < '2022-10-01'
GROUP BY a.iban
ORDER BY
    CASE a.iban
        WHEN 'BG40 RFFX 4898 53DD CZD6 KQ' THEN 1
        WHEN 'PT42 5267 0592 8451 8001 2180 3' THEN 2
        WHEN 'FR96 8758 8909 81LR DJ71 ERKN D56' THEN 3
    END;
```

--14---Top Wishlist Products summary

```
SELECT TOP 3
p.name,
       p.price,
       COUNT(w.product_id) AS total_wishes
FROM products p
JOIN wishlists w ON p.id = w.product_id
WHERE p.in_stock = 1
GROUP BY p.id, p.name, p.price
ORDER BY total_wishes DESC, p.name ASC;
```

--15---E-Commerce Customer Purchases Report

```
SELECT
    c.email,
    COUNT(p.amount) AS total_purchases,
    CAST(SUM(p.amount) AS DECIMAL(10,2)) AS total_purchase_amount
 FROM customers c
 JOIN purchases p ON c.id = p.customer_id
 WHERE p.dt >= '2024-03-01' AND p.dt < '2024-04-01'  -- Only March 2024
 GROUP BY c.email
 ORDER BY c.email ASC;
```

--16—Report on Application Pending Consular Service

```
SELECT
    a.email,
    ap.dt AS scheduled_appointment,
    DATEDIFF(DAY, ap.dt, '2024-04-10') AS days_of_delay
 FROM applicants a
 JOIN appointments ap ON a.id = ap.applicant_id
 WHERE ap.is_received = 0
   AND ap.dt < '2024-04-10'
 ORDER BY ap.dt ASC, a.email ASC;
```

---17---Weekend application for consular

```
SELECT
    a.email,
    DATENAME(WEEKDAY, ap.dt) AS scheduled_appointment
 FROM applicants a
 JOIN appointments ap ON a.id = ap.applicant_id
 WHERE DATEPART(WEEKDAY, ap.dt) IN (1, 7)
 ORDER BY a.email ASC;
```

---18---Active domains Registration by country with tools

```
SELECT
    c.name AS country_name,
    COUNT(d.name) AS total_domains
 FROM countries c
 JOIN domains d
    ON c.id = d.country_id
 WHERE d.is_active = 1
 GROUP BY c.name
 ORDER BY c.name;
```

---19---domain renewal overview

```
 DECLARE @today_date DATE = '2024-04-10';

 SELECT
    name,
    CONVERT(VARCHAR, @today_date, 23) AS today_date,
    CONVERT(VARCHAR, next_renewal_date, 23) AS next_renewal_date,
    DATEDIFF(DAY, @today_date, next_renewal_date) AS days_until_renewal
FROM domains
ORDER BY
    DATEDIFF(DAY, @today_date, next_renewal_date),
    name;
```

--20—User transaction details

```
SELECT
   u.email,
   COUNT(t.amount) AS total_transactions,
   FORMAT(MIN(t.amount), 2) AS min_amount,
   FORMAT(MAX(t.amount), 2) AS max_amount,
   FORMAT(SUM(t.amount), 2) AS total_amount
FROM users u
JOIN transactions t ON u.id = t.user_id
WHERE t.dt LIKE '2024-03-%'
GROUP BY u.email
ORDER BY u.email ASC;
```

21- Total Transactions and sum for each user

```
SELECT
    u.email,
    COUNT(*) AS total_transactions,
    FORMAT(SUM(t.amount), 2) AS total_amount
 FROM users u
 JOIN transactions t ON u.id = t.user_id
 WHERE t.dt LIKE '2023%'
 GROUP BY u.email
 ORDER BY u.email ASC;
```

22-Top Cryptocurrencies by average transaction amount

```
SELECT
    c.name,
    ROUND(AVG(t.amount), 2) AS avg_transaction_amount
 FROM coins c
 JOIN transactions t ON c.id = t.coin_id
 WHERE t.dt LIKE '2023%'
 GROUP BY c.name
 ORDER BY avg_transaction_amount ASC
 LIMIT 3;
```

23-cryptocurrency transactions summery report

```
SELECT
    c.name,
    COUNT(*) AS total_transactions,
    FORMAT(MIN(t.amount), 2) AS min_amount,
    FORMAT(MAX(t.amount), 2) AS max_amount,
    FORMAT(ROUND(AVG(t.amount), 2), 2) AS avg_amount
 FROM coins c
 JOIN transactions t ON c.id = t.coin_id
 WHERE t.dt LIKE '2024-03%'
 GROUP BY c.name
 ORDER BY total_transactions DESC, c.name ASC;
```

24-Antivirus suspicius File extentions report

```
SELECT
    extension,
    COUNT(*) AS total_suspicious_files
```

```
 FROM suspicious_files
 WHERE is_suspicious = 1
   AND scan_dt LIKE '2024-03%'
 GROUP BY extension
 ORDER BY total_suspicious_files DESC, extension ASC
 LIMIT 5;
```

25-antivirus scanned devices report

```
SELECT
    c.email,
    COUNT(d.mac_address) AS total_scanned_devices
 FROM clients c
 JOIN devices d ON c.id = d.client_id
 WHERE d.is_scanned = 1
   AND d.scheduled_scan_dt >= '2024-03-01'
   AND d.scheduled_scan_dt < '2024-04-01'
 GROUP BY c.email
 ORDER BY c.email;
```

26-resource usage report for online hosting panel

```
SELECT
    c.email,
    ROUND(AVG(s.cpu_usage), 2) AS average_cpu_usage,
    ROUND(AVG(s.memory_usage), 2) AS average_memory_usage,
    ROUND(AVG(s.disk_usage), 2) AS average_disk_usage
 FROM customers c
 JOIN site_metrics s ON c.id = s.customer_id
 GROUP BY c.email
 HAVING
    AVG(s.cpu_usage) > 50
    OR AVG(s.memory_usage) > 50
    OR AVG(s.disk_usage) > 50
 ORDER BY c.email;
```

27-dashboard report for online hosting customers panel----------------

```
SELECT c.email, COUNT(s.url) AS total_active_sites

 FROM customer c LEFT JOIN site s ON c.id = s.customer_id

WHERE s.is_active = 1

GROUP BY c.email

ORDER BY c.email ASC;
```

28-average income report in online tax appilication

```
SELECT
    a.iban,
    ROUND(AVG(i.amount), 2) AS average_income,
    ROUND(SUM(i.amount), 2) AS total_income
 FROM accounts a
 JOIN income i ON a.id = i.account_id
 WHERE i.dt >= '2024-01-01' AND i.dt < '2024-04-01'
 GROUP BY a.iban
 ORDER BY average_income DESC, a.iban ASC
 LIMIT 3;
```

29-tax calculation for online tax appilication

```
SELECT
    a.iban,
    ROUND(SUM(i.amount), 2) AS total_income,
    '20%' AS tax_rate,
    ROUND(SUM(i.amount) * 0.20, 2) AS calculated_tax
 FROM accounts a
 JOIN income i ON a.id = i.account_id
 WHERE i.dt >= '2023-01-01' AND i.dt <= '2023-12-31'
 GROUP BY a.iban
 ORDER BY a.iban ASC;
```

30-monthly budget report for online budgeting appilication --

```
SELECT

    c.email,

    COALESCE(ROUND(SUM(e.amount), 2), 0) AS total_expenses,

    COALESCE(ROUND(SUM(i.amount), 2), 0) AS total_income

FROM customers c

LEFT JOIN expenses e

    ON c.id = e.customer_id

   AND e.dt LIKE '2024-03%'

LEFT JOIN income i

    ON c.id = i.customer_id

   AND i.dt LIKE '2024-03%'

GROUP BY c.email

ORDER BY c.email;
```

--31- Balance report for online budgeting application

```
SELECT
     c.email,
     ROUND(COALESCE(i.total_income, 0) - COALESCE(e.total_expenses, 0), 2) AS balance
 FROM customers c
 LEFT JOIN (
     SELECT customer_id, SUM(amount) AS total_income
     FROM income
```

```
    GROUP BY customer_id
) i ON i.customer_id = c.id
LEFT JOIN (
    SELECT customer_id, SUM(amount) AS total_expenses
    FROM expenses
    GROUP BY customer_id
) e ON e.customer_id = c.id
WHERE COALESCE(i.total_income, 0) - COALESCE(e.total_expenses, 0) < 0
ORDER BY c.email;
```

32- Monthly sales report

```
SELECT p.name, MONTHNAME(STR_TO_DATE(s.dt, '%Y-%m-%d %H:%i:%s')) AS month,
ROUND(SUM(s.amount), 2) AS total_sales

FROM products p JOIN sales s ON p.id = s.product_id

WHERE s.dt >= '2024-01-01' AND s.dt < '2024-04-01'

GROUP BY p.name, MONTHNAME(STR_TO_DATE(s.dt, '%Y-%m-%d %H:%i:%s')),
MONTH(STR_TO_DATE(s.dt, '%Y-%m-%d %H:%i:%s'))

 ORDER BY MONTH(STR_TO_DATE(s.dt, '%Y-%m-%d %H:%i:%s')) ASC, total_sales DESC;
```

33- IT project resource analysis

```
SELECT
    p.name AS project_name,
    COUNT(pe.employee_id) AS employee_count,
    CEIL(AVG(e.experience_years)) AS avg_experience_years,
    CASE
       WHEN COUNT(pe.employee_id) < 5 THEN 'Yes'
```

```
        ELSE 'No'
    END AS is_understaffed
 FROM projects p
 JOIN projects_employees pe ON p.id = pe.project_id
 JOIN employees e ON pe.employee_id = e.id
 GROUP BY p.id, p.name
 HAVING AVG(e.experience_years) > 2
 ORDER BY employee_count DESC, project_name ASC;
```

34- Ethereum market dashboard analysis

```
SELECT
    wallet,
    COUNT(*) AS total_transactions,
    ROUND(SUM(CASE WHEN amount > 0 THEN amount ELSE 0 END), 2) AS total_bought,
    ROUND(ABS(SUM(CASE WHEN amount < 0 THEN amount ELSE 0 END)), 2) AS total_sold
 FROM transactions
 WHERE dt LIKE '2024-02%'  -- Only February 2024 transactions
 GROUP BY wallet
 ORDER BY wallet ASC;
```

35- Employee leave tracker

```
SELECT
    e.email,
    COALESCE(SUM(l.days_taken), 0) AS leave_days_taken,
    CASE
        WHEN COALESCE(SUM(l.days_taken), 0) <= 20 THEN 'Within Limit'
        ELSE 'Exceeded'
    END AS leave_status
 FROM employees e
 LEFT JOIN leave_records l
    ON e.id = l.employee_id
   AND l.leave_dt LIKE '2023%'   -- Only records from 2023
 GROUP BY e.email
 ORDER BY e.email ASC;
```

36- Email platform engagement stats

```
SELECT
    c.name,
    SUM(es.emails_sent) AS total_emails_sent,
    SUM(es.emails_opened) AS total_emails_opened,
    ROUND(SUM(es.emails_opened) * 100.0 / SUM(es.emails_sent), 2) AS open_rate
 FROM campaigns c
 JOIN email_stats es ON c.id = es.campaign_id
 GROUP BY c.id, c.name
 HAVING ROUND(SUM(es.emails_opened) * 100.0 / SUM(es.emails_sent), 2) > 50
 ORDER BY open_rate DESC, c.name ASC;
```

37- Bond maturity analysis

```
SELECT
    b.name,
    COUNT(m.maturity) AS maturity_dates,
    MIN(m.maturity) AS earliest_maturity,
    MAX(m.maturity) AS latest_maturity,
    CEIL(AVG(DATEDIFF(m.maturity, '2023-09-13'))) AS avg_days_to_maturity
 FROM bonds b
 JOIN maturities m ON b.id = m.bond_id
 GROUP BY b.id, b.name
 HAVING CEIL(AVG(DATEDIFF(m.maturity, '2023-09-13'))) > 365
 ORDER BY b.name ASC;
```

38- Bond interest rate analysis

```
SELECT
    b.name,
    COUNT(ir.rate) AS interest_rates,
    CAST(MIN(ir.rate) AS DECIMAL(3,1)) AS lowest_rate,
    CAST(MAX(ir.rate) AS DECIMAL(3,1)) AS highest_rate,
    FORMAT(AVG(ir.rate), 2) AS avg_rate
 FROM bonds b
 JOIN interest_rates ir ON b.id = ir.bond_id
 GROUP BY b.id, b.name
 HAVING AVG(ir.rate) > 3
 ORDER BY b.name ASC;
```

39- Bond cash flow analysis for bondholders

```
SELECT
    bh.name,
    FORMAT(SUM(b.annual_coupon * b.coupons_remaining), 2) AS total_cash_flow
 FROM bondholders bh
 JOIN bondholders_bonds bhb ON bh.id = bhb.bondholder_id
 JOIN bonds b ON bhb.bond_id = b.id
 GROUP BY bh.id, bh.name
 HAVING SUM(b.annual_coupon * b.coupons_remaining) > 10000
 ORDER BY total_cash_flow DESC;
```

40- Sum of the cash flows analysis

```
SELECT
    i.email,
    COUNT(c.cash_flow) AS investments,
    MIN(c.cash_flow) AS min_cash_flow,
    MAX(c.cash_flow) AS max_cash_flow,
    ROUND(AVG(c.cash_flow), 2) AS avg_cash_flow
```

```
 FROM investors i
 JOIN cash_flows c ON i.id = c.investor_id
 GROUP BY i.id, i.email
 HAVING SUM(c.cash_flow) > 1000000
 ORDER BY i.email ASC;
```

(41) Expected Cash Flow Analysis

```
 SELECT i.email, COUNT(cf.expected_flow)

AS investment_count,

SUM(cf.expected_flow) AS total_expected_flow,

 (MAX(cf.expected_flow) - MIN(cf.expected_flow)) AS range_expected_flow

 FROM investors JOIN cash_flows cf ON i.id = cf.investor_id

GROUP BY i.email HAVING SUM(cf.expected_flow) > 100000

ORDER BY i.email ASC;
```

(42) online store coupen codes report

```
 SELECT c.coupon_code, c.description, COUNT(cu.amount) AS total_uses, MIN(cu.amount) AS
min_discount, MAX(cu.amount) AS max_discount,

 ROUND(AVG(cu.amount), 2) AS avg_discount

 FROM coupons c JOIN coupon_uses cu ON c.id = cu.coupon_id

WHERE c.is_enabled = 1

GROUP BY c.coupon_code, c.description ORDER BY c.coupon_code;
```

(43) freelancer platform yearly inocome Report

```
WITH completed_projects AS ( SELECT p.id, f.profession_id, p.income
FROM projects p JOIN freelancers f ON p.freelancer_id = f.id

WHERE p.status = 'Completed' )

SELECT pr.title, COUNT(cp.id) AS total_projects, ROUND(SUM(cp.income), 2) AS total_income,
COUNT(DISTINCT f.id) AS total_freelancers, ROUND(SUM(cp.income)/NULLIF(COUNT(DISTINCT
f.id),0), 2) AS average_income_per_freelancer

FROM completed_projects cp

JOIN professions pr ON cp.profession_id = pr.id

JOIN freelancers f ON cp.profession_id = f.profession_id

GROUP BY pr.title ORDER BY total_income DESC;
```

(44) Ecommerce Warehouse stock report

```
SELECT c.title AS category, p.title, SUM(p.stock_number) AS total_stock

FROM products p JOIN categories c ON p.category_id = c.id

GROUP BY c.title, p.title HAVING SUM(p.stock_number) > 10

ORDER BY c.title ASC, p.title ASC, total_stock DESC;
```

45) Antivirus Database quarantine Report

```
 SELECT domain_name, tt.threat_type, COUNT(*) AS total_occurrences, SUM(users_affected) AS
total_users_affected

FROM quarantine_urls qu JOIN threat_types tt ON qu.threat_id = tt.id

WHERE qu.status = 'Quarantined' GROUP BY domain_name, tt.threat_type

ORDER BY total_users_affected DESC, domain_name ASC;
```

(46) Online Streaming Service traffice report

```
 SELECT c.mac_address, COUNT(*) AS streams, SUM(s.traffic) AS total_traffic

FROM streams s JOIN clients c ON s.client_id = c.id

WHERE s.quality IN ('720p', '1080p', '1440p', '2160p') GROUP BY c.mac_address ORDER BY total_traffic DESC;
```

(47) Cloud hosting instances Performance statistics

```
SELECT n.cidr, COUNT(*) AS instances, CONCAT(CEILING(AVG(CAST(REPLACE(i.cpu_usage, '%', '') AS DECIMAL(5,2)))), '%') AS avg_cpu_usage, CONCAT(CEILING(AVG(CAST(REPLACE(i.memory_usage, '%', '') AS DECIMAL(5,2)))), '%') AS avg_memory_usage, CONCAT(CEILING(AVG(CAST(REPLACE(i.network_usage, '%', '') AS DECIMAL(5,2)))), '%') AS avg_network_usage

FROM networks n JOIN instances i ON n.id = i.network_id

WHERE CAST(REPLACE(i.cpu_usage, '%', '') AS DECIMAL(5,2)) >= 80 GROUP BY n.cidr ORDER BY n.cidr;
```

(48) AI VIdeo Processing Service Usage Time Calculation

```
 SELECT t.hash, SUM(DATEDIFF(SECOND, p.start_dt, p.end_dt)) AS usage_time

 FROM tasks t JOIN processes p ON t.id = p.task_id

 GROUP BY t.hash

ORDER BY usage_time DESC;
```

(49) Benchmarking Tool Report

```
SELECT CONCAT('Device ', id, ' has class: ',

CASE WHEN score >= 80 THEN 'A'
```

```
 WHEN score >= 60 THEN 'B' WHEN score >= 40 THEN 'C' WHEN score >= 20

THEN 'D' ELSE 'F' END) AS device FROM devices ORDER BY id;
```

(50) Smart home Application customer Report

```
SELECT a.username, a.email, MAX(t.name) AS highest_tariff, SUM(r.amount) AS consumption,
ROUND(SUM(r.amount * t.cost), 2) AS total_cost

FROM accounts a JOIN readings r ON a.id = r.account_id

JOIN tariffs t ON r.tariff_id = t.id

GROUP BY a.username, a.email

 ORDER BY a.username ASC
```

Answer 51: MMORG Game Inventory Overload Notification

```
SELECT a.username, a.email, COUNT(ai.item_id) AS items, SUM(i.weight) AS total_weight

FROM accounts a JOIN accounts_items ai ON a.id = ai.account_id

JOIN items i ON ai.item_id = i.id

GROUP BY a.id, a.username, a.email

HAVING SUM(i.weight) > 20

ORDER BY total_weight DESC, a.username ASC;
```

Answer 52: Outdoor Banner MarketPlace Placement Report

```
SELECT c.name AS city, COUNT(*) AS banners,

MIN(b.width * b.height) AS min_area,

CEILING(AVG(b.width * b.height)) AS avg_area,

MAX(b.width * b.height) AS max_area,

SUM(b.width * b.height) AS total_area
```

```
FROM banners b JOIN cities c ON b.city_id = c.id

GROUP BY c.name ORDER BY c.name;
```

Answer 53: Auction web service lot statistics

```
SELECT l.name, COUNT(o.amount) AS offers,

CASE WHEN COUNT(o.amount) = 0

 THEN NULL ELSE CAST(ROUND(MIN(o.amount), 2) AS DECIMAL(10,2)) END AS min_offer,

 CASE WHEN COUNT(o.amount) = 0 THEN NULL

ELSE CAST(ROUND(AVG(o.amount), 2) AS DECIMAL(10,2)) END AS avg_offer,

 CASE WHEN COUNT(o.amount) = 0 THEN NULL

 ELSE CAST(ROUND(MAX(o.amount), 2) AS DECIMAL(10,2)) END AS max_offer

FROM lots l LEFT JOIN offers o ON l.id = o.lot_id GROUP BY l.id, l.name ORDER BY offers DESC;
```

```
SELECT

    l.name,

    COUNT(o.amount) AS offers,

    ROUND(MIN(o.amount), 2) AS min_offer,

    ROUND(AVG(o.amount), 2) AS avg_offer,

    ROUND(MAX(o.amount), 2) AS max_offer

FROM

    lots l

LEFT JOIN

    offers o ON l.id = o.lot_id
```

```
GROUP BY
    l.id, l.name
ORDER BY
    offers DESC;
```

Answer 54: The calculator web service simple Report

```
SELECT CONCAT(a.last_name, ' ', a.first_name) AS full_name, a.iban,
 CAST(SUM(d.income) AS DECIMAL(12,2)) AS income, '10%' AS rate,
CAST(ROUND(SUM(d.income) * 0.10, 2) AS DECIMAL(12,2)) AS tax
FROM accounts a JOIN declarations d ON a.id = d.account_id
GROUP BY a.id, a.last_name, a.first_name, a.iban
ORDER BY full_name ASC;
```

Answer 55: Social Network Relationship statistics

```
SELECT CONCAT(p.last_name, ' ', p.first_name) AS full_name, p.email, COUNT(r.profile_id) AS total_relations,
 SUM(CASE WHEN r.is_approved = 1 THEN 1 ELSE 0 END) AS approved_relations,
SUM(CASE WHEN r.is_approved = 0 THEN 1 ELSE 0 END) AS pending_relations
 FROM profiles p LEFT JOIN relations r ON p.id = r.profile_id
GROUP BY p.id, p.last_name, p.first_name, p.email
ORDER BY full_name ASC;
```

Answer 56: Online banking Transactions

```
SELECT a.iban, COUNT(t.amount) AS transactions,
CAST(ROUND(SUM(t.amount), 2) AS DECIMAL(12,2)) AS total
```

```
FROM accounts a

JOIN transactions t ON a.id = t.account_id WHERE YEAR(t.dt) = 2022 AND MONTH(t.dt) = 9

GROUP BY a.iban ORDER BY total DESC;
```

Answer 57: Internet Service Provider monthly Report

```
SELECT c.mac, SUM(t.amount) AS traffic,

CAST(ROUND(SUM(t.amount) * c.tariff, 2) AS DECIMAL(10,2)) AS cost

 FROM clients c JOIN traffic t ON c.id = t.client_id

WHERE t.dt LIKE '2022-05%'

GROUP BY c.mac, c.tariff ORDER BY cost DESC;
```

Answer 58: The Yellow Pages Companies Report

```
SELECT c.name, c.address, c.phone,

 CONCAT(CAST(ROUND(AVG(cat.review_rating*1.0), 1) AS VARCHAR(10)), ' (', COUNT(cat.name), ' categories)') AS overall_review_rating

FROM companies c JOIN categories cat ON c.id = cat.company_id

 GROUP BY c.id, c.name, c.address, c.phone

 ORDER BY ROUND(AVG(cat.review_rating*1.0), 1) DESC, c.name ASC;
```

Answer 59: Domain Name Registar Accounts Report

```
SELECT a.username, COUNT(d.name) AS domains, MIN(d.expiration_date) AS nearest_expiration

FROM accounts a JOIN domains d ON a.id = d.account_id

 WHERE a.is_active = 1 AND d.expiration_date > '2022-07-15'

GROUP BY a.username ORDER BY a.username ASC;
```

Answer 60: Advertising Network Events Report

```
SELECT c.name AS campaign, COUNT(e.value) AS events, FORMAT(AVG(e.value), 'N5') AS average_value
```

```
FROM campaigns c JOIN events e ON c.id = e.campaign_id

WHERE LEFT(e.dt, 10) = '2022-07-15'

GROUP BY c.name HAVING AVG(e.value) >= 0.7

ORDER BY average_value DESC;
```

61. Ecommerce deal report

```
SELECT TOP 3 p.first_name, p.last_name,

SUM(d.amount) AS total_amount FROM

 profile p JOIN deals d ON p.id = d.profile_id

GROUP BY p.id, p.first_name, p.last_name

ORDER BY total_amount DESC;
```

62. Freelance platform candidate report

```
SELECT TOP(10) p.first_name, p.last_name, p.email, s.job_success_score

FROM profiles p JOIN stats s ON p.id = s.profile_id

WHERE p.is_verified = 1 AND s.job_success_score >= 90

ORDER BY s.job_success_score DESC, p.first_name ASC, p.last_name ASC;
```

63.virtual machine deployment report

```
 SELECT c.name AS configuration,

   COUNT(*) AS deployments FROM configurations AS c

JOIN deployments AS d  ON c.id = d.configuration_id

WHERE YEAR(CONVERT(DATE, d.dt)) = 2021GROUP BY c.nameORDER BY COUNT(*) DESC;
```

64. Visitors behaviour report

```
SELECT COUNT(*) AS purchases
```

```
FROM events

WHERE type = 'buy'

  AND dt >= '2022-05-01'

  AND dt <  '2022-06-01';
```

65. Advertising campaigns report

```
SELECT    c.name AS company_name,

    SUM(ca.revenue - ca.expenses) AS profit

FROM  companies c

JOIN  campaigns ca ON c.id = ca.company_id

GROUP BY c.name

HAVING  SUM(ca.revenue - ca.expenses) > 0

ORDER BY

    profit DESC

LIMIT 3;
```

66.Traffic Audit Report

```
 SELECT mac, upstream_rate, downstream_rate, downtime_rate

FROM clients

 WHERE downstream_rate > upstream_rate AND (downtime_rate = 'never' OR downtime_rate = 'once')

ORDER BY mac ASC;
```

67. Calender Application events Reports

```
SELECT
   e.dt,
   e.title,
   o.full_name,
   o.email_address
FROM event e
JOIN owner o ON e.owner_id = o.id
WHERE o.on_vacation = 0 ORDER BY  e.dt ASC LIMIT 5;
```

68. Fire Wall Active Clients Tracking

```
SELECT DISTINCT c.mac
FROM clients c JOIN traffic t ON c.id = t.client_id
ORDER BY  c.mac ASC;
```

69. Active Wallets

```
SELECT w.address
FROM wallets w
JOIN transactions t ON w.id = t.wallet_id
GROUP BY w.address;
```

70. Animal Tracking

```
SELECT a.name
FROM animals a
JOIN tracklog t ON a.id = t.animal_id
```

```
GROUP BY a.name

order by a.name asc;
```

71. Exchange Rates

```
 SELECT c.customer_name, ROUND(SUM(CASE WHEN o.order_type = 'Buy'

 THEN o.order_amount * 0.001 ELSE 0 END) + SUM(CASE WHEN o.order_type = 'Sell'

THEN o.order_amount * 0.0015 ELSE 0 END), 2) AS total_fees

FROM Customers AS c JOIN Orders AS o ON c.id = o.customer_id

GROUP BY c.customer_name

ORDER BY c.customer_name;
```

72. Credit dues

```
SELECT ch.first_name +' '+ ch.last_name AS full_name,

ROUND(SUM(t.amount) * ( 1 + ch.interest_rate / 100 ), 2)

AS dues FROM Credit_Holders AS ch JOIN Transactions AS t

ON ch.id = t.credit_holder_id

WHERE ch.interest_rate > 12

 GROUP BY ch.first_name,ch.last_name,ch.interest_rate

ORDER BY dues DESC;
```

73. Interest earned

```
SELECT

  account_holder,

  CONCAT( LEFT(amount, 1), FORMAT(CAST(SUBSTRING(amount, 2) AS DECIMAL
```

(10,2)) * 0.05, 2)) As interest

FROM accounts

ORDER BY account_holder asc;

74. Monthly revenue

```
SELECT
  SUBSTRING(transaction_id, 1, 2) AS year,
  SUBSTRING(transaction_id, 3, 3) AS month,
  ROUND(SUM(amount), 2) AS total_transactions
FROM transactions
GROUP BY year, month ORDER BY year, month;
```

75. Final result

```
SELECT CONCAT(first_name, ' ', last_name) AS full_name,
ROUND((cgpa_first_year + cgpa_second_year + cgpa_third_year + cgpa_fourth_year) / 4, 1) AS average_gpa
FROM results ORDER BY average_gpa DESC;
```

76. Mutual funds

```
SELECT MONTH(order_date) AS month, fund_name, SUM(order_amount) AS total_investments
FROM funds GROUP BY month, fund_name
ORDER BY month, fund_name;
```

77.DPI software protocols report

```
SELECT protocol, SUM(traffic_in) AS traffic_in, SUM(traffic_out) AS traffic_out FROM traffic GROUP BY
protocol HAVING SUM(traffic_in) > SUM(traffic_out) ORDER BY protocol ASC;
```

78. Advertising system failure reports

```
SELECT CONCAT(c.first_name, ' ', c.last_name) AS customer,

COUNT(*) AS failures FROM customers c JOIN campaigns ca

ON c.id = ca.customer_id JOIN events e ON ca.id = e.campaign_id

WHERE e.status = 'failure' GROUP BY c.id HAVING failures > 3;
```

79. Election Exit poll report

```
SELECT CONCAT(c.first_name, ' ', c.last_name) AS candidate,

 COUNT(r.vote_at) AS votes FROM candidates c JOIN results r

ON c.id = r.candidate_id GROUP BY c.id

 ORDER BY votes DESC, candidate ASC;
```

80. Billing anlytics customer report

```
SELECT customer, COUNT(*) AS transactions, ROUND(SUM(amount), 2) AS total

FROM events

WHERE dt LIKE '2021-12%'

GROUP BY customer

HAVING transactions >= 3

ORDER BY customer;
```

81 Aggregate Marks

```
SELECT STUDENT_ID, SUM(MARKS) AS SUM_OF_MARKS

 FROM marks GROUP BY STUDENT_ID HAVING SUM(MARKS) >= 500

 ORDER BY STUDENT_ID DESC;
```

82 Trip query

```
SELECT MAX(cnt)

FROM ( SELECT f.ID, COUNT(*) AS cnt

FROM families f JOIN countries c ON f.FAMILY_SIZE >= c.MIN_SIZE GROUP BY f.ID ) t;
```

83 Activity query

```
SELECT ACTIVITY

 FROM friends GROUP BY ACTIVITY HAVING COUNT() NOT IN ( SELECT MAX(cnt)

 FROM (SELECT COUNT() AS cnt

FROM friends GROUP BY ACTIVITY) t UNION SELECT MIN(cnt)

 FROM (SELECT COUNT(*) AS cnt FROM friends GROUP BY ACTIVITY) t );
```

84 Restaurants groups

```
SELECT visited_on, amount, ROUND( ( SELECT SUM(amount) FROM customers c2 WHERE c2.visited_on
BETWEEN c1.visited_on-INTERVAL 6 DAY AND c1.visited_on ) / ( SELECT COUNT(*) FROM customers c2
WHERE c2.visited_on BETWEEN c1.visited_on-INTERVAL 6 DAY AND c1.visited_on ), 0 ) AS avg_amount FROM
customers c1 ORDER BY visited_on;
```

85 Examination data management

```
SELECT STUDENT_ID, SUBJECT, COUNT(*) AS NUMBER_OF_TIMES FROM examination GROUP BY
STUDENT_ID, SUBJECT;
```

86 the perfect arrangement

```
SELECT ID, FIRST_NAME, LAST_NAME FROM customer WHERE LENGTH(CONCAT(FIRST_NAME, LAST_NAME))
< 12 ORDER BY LENGTH(CONCAT(FIRST_NAME, LAST_NAME)), LOWER(CONCAT(FIRST_NAME, LAST_NAME)),
ID;
```

87 students score

```
SELECT ID, NAME

FROM student
```

```
WHERE SCORE > (SELECT AVG(SCORE) FROM student)

ORDER BY ID;
```

88 the first orders

```
SELECT id, order_date, status, customer_id FROM orders WHERE status <> 'DELIVERED' ORDER BY
order_date ASC, id ASC LIMIT 5;
```

89 customers credit limit

```
SELECT ID, NAME

FROM customer

WHERE COUNTRY = 'USA' AND CREDITS > 100000 ORDER BY ID ASC;
```

90 The beautiful collection

```
SELECT  CASE

    WHEN RED = GREEN AND GREEN = BLUE THEN 'GOOD'

    WHEN RED = GREEN OR RED = BLUE OR GREEN = BLUE THEN 'BAD'

    ELSE 'WORSE'

  END AS TYPE_OF_COLLECTION

FROM collection;
```

91. Big Companies

```
SELECT  C.NAME

FROM

    COMPANY C

JOIN

    SALARY S ON C.ID = S.COMPANY_ID
```

```
GROUP BY
   C.ID, C.NAME HAVING
   AVG(S.SALARY) > 40000;
```

92. Scheduling errors

```
SELECT DISTINCT
   P.NAME AS professor_name,
   C.NAME AS course_name
FROM
   PROFESSOR P
JOIN
   SCHEDULE S ON P.ID = S.PROFESSOR_ID
JOIN
   COURSE C ON S.COURSE_ID = C.ID
WHERE
   P.DEPARTMENT_ID != C.DEPARTMENT_ID;
```

93 . List the course names

```
SELECT DISTINCT
   P.NAME AS professor_name,
   C.NAME AS course_name
```

```
FROM PROFESSOR P

JOIN SCHEDULE S ON P.ID = S.PROFESSOR_ID

JOIN COURSE C ON S.COURSE_ID = C.ID;
```

94. Professor names and salaries

```
SELECT P.NAME,P.SALARY

FROM PROFESSOR P

JOIN DEPARTMENT D ON P.DEPARTMENT_ID = D.ID

WHERE  D.NAME != 'Arts and Humanities'

    AND P.SALARY > (

        SELECT MIN(P2.SALARY)

        FROM PROFESSOR P2

        JOIN DEPARTMENT D2 ON P2.DEPARTMENT_ID = D2.ID WHERE D2.NAME = 'Arts and Humanities' );
```

95. Students major

```
SELECT S.STUDENT_NAME, M.MAJOR_NAME FROM STUDENTS S JOIN REGISTER R ON S.STUDENT_ID =
R.STUDENT_ID JOIN MAJORS M ON R.MAJOR_ID = M.MAJOR_ID ORDER BY S.STUDENT_ID LIMIT 20;
```

96. Student Rank

```
 SELECT  SCORE

FROM  STUDENT

ORDER BY SCORE DESC

LIMIT 1 OFFSET 212;
```

97. Clumsy administrator

```
SELECT DISTINCT NAME
```

```
FROM EMPLOYEE

GROUP BY NAME, PHONE, AGE

HAVING COUNT(*) > 1;
```

98.Accounting software balance report

```
SELECT customer, FORMAT(SUM(debit) - SUM(credit), 2) AS balance FROM transactions

 WHERE DATE_FORMAT(STR_TO_DATE(dt, '%Y-%m-%d  %H:%i:%s'), '%Y-%m') = '2021-12'

GROUP BY customer

 ORDER BY customer ASC;
```