Part 2: Database

Task 1.1:

When an UPDATE statement is mistakenly executed in production without a WHERE clause such as setting all email_verified statuses to 1 and no recent backup is available, the database can still be recovered using one of two main approaches:

1. Recovery Using Binary Logs (Binlogs)
2. Reverse Audit Recovery Plan

#Recovery Using Binary Logs (Binlogs):
If MySQL binary logging is enabled and the binlog_format is set to ROW or MIXED, it is possible to recover the original data using the mysqlbinlog utility.

First, determine the approximate time the erroneous query was executed. Then extract the relevant portion of the binary log using the following command:

***mysqlbinlog --start-datetime="2025-05-03 13:00:00"* **
***--stop-datetime="2025-05-03 13:15:00"* **
/var/log/mysql/mysql-bin.000001 > binlog_recovery.sql

This command exports all queries executed within the specified time window to a file (binlog_recovery.sql). Upon reviewing this file, locate the faulty UPDATE statement. If the binary log format is ROW, the file will include both the before and after values of each affected row.

Using the "before" values, generate reverse UPDATE statements to restore the original state.

Example:

UPDATE users SET email_verified = 0 WHERE id = 101;
UPDATE users SET email_verified = 0 WHERE id = 102;
—

#Reverse Audit Recovery Plan :
If binary logging is disabled, recovery depends on alternate sources such as application-level audit logs or external snapshots.

If the application maintains a change history or audit trail (e.g., a user_audits table), retrieve the original values from there and restore them using UPDATE statements.

Task 1.2:

If the system tracks changes via audit tables, retrieve previous email_verified values for each user and reverse them

Example Query :

UPDATE users u
JOIN user_audits a ON u.id = a.user_id
SET u.email_verified = a.previous_value
WHERE a.field_changed = 'email_verified' AND a.changed_at BETWEEN '2025-05-03 13:00:00' AND '2025-05-03 13:15:00';

Task: 2

```
use Illuminate\Support\Facades\DB;

function compareCampaignSpending()
{
   // Fetch data from MySQL (application DB)
   $mysqlCampaigns = DB::table('campaigns')
      ->select('campaigncode', 'user_id', DB::raw('fixed_price * campaignshow as calculated_spend'))
      ->get()
      ->keyBy('campaigncode'); // or use ->keyBy('user_id') based on need

   // Fetch data from MSSQL (finance DB)
   $mssqlInvoiced = DB::connection('sqlsrv')
      ->table('invoices')
      ->select('campaigncode', 'user_id', 'billed_amount')
      ->get()
      ->keyBy('campaigncode'); // or ->keyBy('user_id')

   $mismatches = [];

   // Compare both datasets
   foreach ($mysqlCampaigns as $code => $campaign) {
      if (!isset($mssqlInvoiced[$code])) {
```

```
            $mismatches[] = [
                'campaigncode' => $code,
                'issue' => 'Missing in MSSQL',
                'calculated_spend' => $campaign->calculated_spend,
                'billed_amount' => null,
            ];
            continue;
        }

        $invoice = $mssqlInvoiced[$code];

        if (abs($campaign->calculated_spend - $invoice->billed_amount) > 0.01) {
            $mismatches[] = [
                'campaigncode' => $code,
                'user_id' => $campaign->user_id,
                'calculated_spend' => $campaign->calculated_spend,
                'billed_amount' => $invoice->billed_amount,
                'difference' => $campaign->calculated_spend - $invoice->billed_amount,
            ];
        }
    }

    return $mismatches;
}
```

Task: 3.1

SQL query to identify such invalid campaigns:

SELECT *
FROM campaigns
WHERE start_date > end_date;

Task: 3.2

solution using database constraints or application-level validation

ALTER TABLE campaigns
ADD CONSTRAINT chk_date_range CHECK (start_date <= end_date);

Task: 4.1

Campaigns with NULL or non-matching user_id

```
SELECT c.*
FROM campaigns c
LEFT JOIN users u ON c.user_id = u.id
WHERE c.user_id IS NULL OR u.id IS NULL;
```

Task: 4.2

Campaigns with missing campaign_category_id references

```
SELECT c.*
FROM campaigns c
LEFT JOIN campaign_categories cc ON c.campaign_category_id = cc.id
WHERE c.campaign_category_id IS NULL OR cc.id IS NULL;
```

Task: 5

SQL query to return campaign ID, campaign name, and total spend (fixed_price × campaignshow)
per day

```
SELECT
   c.id AS campaign_id,
   c.campaign_name,
   DATE(cs.date) AS spend_date,
   SUM(c.fixed_price * cs.campaignshow) AS total_spend
FROM
   campaigns c
JOIN
   campaign_shows cs ON cs.campaign_id = c.id
WHERE
   c.status = 1
   AND cs.date >= CURDATE() - INTERVAL 7 DAY
GROUP BY
   c.id, c.campaign_name, DATE(cs.date)
ORDER BY
   spend_date DESC, campaign_id;
```

Task: 6

SQL query that returns all such campaigns based on above scenario for a given publishercode = 'PUB123'.

```
SELECT c.*
FROM campaigns c
JOIN campaign_publishers cp ON cp.campaign_id = c.id
WHERE
   c.publisher_selection = 2
   AND cp.publishercode = 'PUB123'
   AND c.created_at >= CURDATE() - INTERVAL 7 DAY
ORDER BY
   c.created_at DESC;
```

Task: 7

query that finds campaigns where current date is between start_date and end_date, and total spend
(fixed_price × campaignshow) is < 50% of total_budget.

```
SELECT
   id AS campaign_id,
   campaign_name,
   fixed_price,
   campaignshow,
   (fixed_price * campaignshow) AS total_spend,
   total_budget,
   start_date,
   end_date
FROM
   campaigns
WHERE
   CURDATE() BETWEEN start_date AND end_date
   AND (fixed_price * campaignshow) < (0.5 * total_budget);
```

Task: 8

query to find all campaigns where daily_budget exceeds total_budget. Then describe how to enforce
this at the DB level and app logic level

```
SELECT
   id AS campaign_id,
   campaign_name,
   daily_budget,
   total_budget
FROM
   campaigns
WHERE
   daily_budget > total_budget;
```

Task: 9

query that finds overlapping campaigns for the same user_id and same campaign_type.

```
SELECT
   c1.id AS campaign_id_1,
   c2.id AS campaign_id_2,
   c1.user_id,
   c1.campaign_type,
   c1.start_date AS start_1,
   c1.end_date AS end_1,
   c2.start_date AS start_2,
   c2.end_date AS end_2
FROM
   campaigns c1
JOIN
   campaigns c2
   ON c1.user_id = c2.user_id
   AND c1.campaign_type = c2.campaign_type
   AND c1.id < c2.id
   AND c1.start_date <= c2.end_date
   AND c1.end_date >= c2.start_date;
```

Part 3: Server Management

Task 1

Deploy Laravel on DigitalOcean with Nginx:

Step 1: Create a DigitalOcean Droplet
→ Login to DigitalOcean.and create droplet

Choose following :

1. Ubuntu 22.04 LTS as OS
2. Basic shared CPU, e.g., 1GB/1vCPU
3. Data center region near your target users
4. Enable SSH or use a password
5. Choose a hostname and create the droplet.

Step 2: Connect via SSH command:
ssh root@your_droplet_ip

Step 3: Update System & Install Essentials

apt update && apt upgrade -y
apt install nginx mysql-server php php-fpm php-mysql php-mbstring php-xml php-curl php-zip php-bcmath php-cli unzip git curl -y
Install Composer

curl -sS https://getcomposer.org/installer | php
mv composer.phar /usr/local/bin/composer

Step 4: Configure MySQL

Mysql_secure_installation

Then create your Laravel database and user:

```
CREATE DATABASE laravel_db;
CREATE USER 'laravel_user'@'localhost' IDENTIFIED BY 'StrongPassword123';
GRANT ALL PRIVILEGES ON laravel_db.* TO 'laravel_user'@'localhost';
FLUSH PRIVILEGES;
```

Step 5: You can clone your Laravel project using Git:

```
cd /var/www/
git clone https://github.com/your/repo.git laravel-app
cd laravel-app
composer install
```

Step 6: Set Permissions

chown -R www-data:www-data /var/www/laravel-app
chmod -R 775 /var/www/laravel-app/storage /var/www/laravel-app/bootstrap/cache

Step 7: Configure .env

Copy example.env file as .env file, configure it as your own needed with databases and other environment .

Step 8: Configure Nginx

nano /etc/nginx/sites-available/laravel

Config code are :

```
server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com;

    root /var/www/laravel-app/public;
    index index.php index.html;

    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-Content-Type-Options "nosniff";

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php8.1-fpm.sock;
    }

    location ~ /\.ht {
        deny all;
    }
```

}

Step 10: (Optional) Install SSL via Let's Encrypt

apt install certbot python3-certbot-nginx -y
certbot --nginx -d yourdomain.com -d www.yourdomain.com

Task 2

Domain Shows Default Nginx Page Instead of Laravel App

Step 1: Disable the default site

```
server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com;
    root /var/www/laravel-app/public;
    ...
}
```

Step 3: Enable the site and reload Nginx

ln -s /etc/nginx/sites-available/laravel /etc/nginx/sites-enabled/
nginx -t
systemctl reload nginx

Step 4: Clear browser or DNS cache if needed

Task 3

If the SSL certificate renewal fails and your website becomes inaccessible via HTTPS, the first step is to verify whether the certificate has actually expired by running sudo certbot certificates. Next, check the renewal logs located at /var/log/letsencrypt/letsencrypt.log to identify the root cause of the failure. Common issues include DNS misconfiguration, missing domain verification files, or changes in server IPs. You can then attempt a manual dry-run renewal using sudo certbot renew --dry-run to simulate the process without making real changes. If that fails, reissue the certificate manually with sudo certbot --nginx -d yourdomain.com -d www.yourdomain.com to re-verify and obtain a new certificate. After renewal, confirm that your Nginx configuration correctly references the certificate and key paths (fullchain.pem and privkey.pem) in the server block for HTTPS. Finally, test the Nginx configuration with nginx -t and reload the service using sudo systemctl reload nginx. Also, ensure your firewall is allowing HTTPS traffic on port 443 with sudo ufw allow 'Nginx Full'. This process should restore secure access to your site via HTTPS.