Business Problem

Build data infrastructure, for e-commerce business to Grow. Requirements will be loading data, automation, security, backup, and data recovery.

Initial Requirements:

- Update Order items table with products table data(product_id)
- Company requires to track whether each item sold is the primary item (first item in shopping cart) or a cross-sold item
- Create an orders Table that captures order_ids with fields; created_at, website_session_id, primary product_id. # of items purchased.
- Need to update orders table, as new orders are added into the order_items table
- Require a website_sessions
    - Aggregate of number of website sessions sliced by year, month, utm_source, utm_campaign
    - Need a page_views table to store page_view data
- Create EER Diagram of the Database
- There are some staff that are not too familiar with SQL, and needs a simply way to query total orders and revenue for a given time period.

Additional Requirements:

- Company wants to start a chat service to give support on their website
- They require data to be stored in the db
- Need to create appropriate tables, and fields.
- Simple command (Stored procedure) to pull count of chats handled by representative(s)
- Additional views for reporting purposes that shows monthly order volyme and revenue, and another that shows monthly website traffic.
    - Require users to be limited to these tools

Results Summary:

- RDBMS used: MySQL
- Data loaded into the db without any errors.
- All key-constraints are holding, and automated triggers are working as intended.
- Chat service tables added in, need to monitor and audit.
- All stored procedures and views are working, and user access restrictions have been implemented.

Table of Contents

Business Problem 1
Initial Data Loading 3
Tracking if product was primary item 4
Creation of orders table 4
    Adding Trigger to update orders table as new order_items are added in 5
Website Sessions 5
    Schema 5
    View 6
Total Orders and Revenue – Stored Procedures 6
Chat Support Data Schema 7
Additional Stored Procedures and Views 9
Final EER Diagram 11
Security Plan Recommendation 12

Initial Data Loading

DDL – Schema Creation

```
CREATE TABLE order_items(
    order_item_id BIGINT,
    created_at DATETIME,
    order_id BIGINT,
    price_usd DECIMAL(6,2),
    cogs_usd DECIMAL(6,2),
    webside_session_id BIGINT,
    PRIMARY KEY(order_item_id)
);
```

```
CREATE TABLE order_item_refunds(
    order_item_refund_id BIGINT,
    created_at DATETIME,
    order_item_id BIGINT,
    order_id BIGINT,
    refund_amount_usd DECIMAL(6,2),
    PRIMARY KEY(order_item_refund_id),
    FOREIGN KEY(order_item_id) REFERENCES order_items(order_item_id)
);
```

```
CREATE TABLE products(
    product_id BIGINT,
    created_at DATETIME,
    product_name VARCHAR(50),
    PRIMARY KEY(product_id)
);
```

- Need to add product table, and backpropagation product_id into order_items
- Only product sold up to the time was product 1, a simple alter table and update will do this.

```sql
ALTER TABLE order_items
ADD COLUMN product_id BIGINT;

UPDATE order_items
SET product_id = 1
WHERE order_item_id > 0;
```

- Adding foreign key mapping to order_items

```sql
ALTER TABLE order_items
ADD CONSTRAINT fk_product_id
FOREIGN KEY(product_id) REFERENCES products(product_id);
```

Tracking if product was primary item

- Since all orders up to this point only has one product, all is_primary is set to 1 for all data

```sql
ALTER TABLE order_items
ADD COLUMN is_primary_item INT;

UPDATE order_items
SET is_primary_item = 1
WHERE order_item_id > 0;
```

Creation of orders table

```sql
CREATE TABLE orders AS
SELECT
    order_id,
    MIN(created_at) AS created_at,
    MIN(webside_session_id) AS website_session_id,
    SUM(CASE
        WHEN is_primary_item = 1 THEN product_id
        ELSE NULL
        END) AS primary_product_id,
    COUNT(order_item_id) AS items_purchased,
    SUM(price_usd) AS price_usd,
    SUM(cogs_usd) AS cogs_usd
FROM order_items
GROUP BY order_id
ORDER BY order_id;
```

```sql
ALTER TABLE orders
ADD CONSTRAINT PRIMARY KEY(order_id);
```

Adding Trigger to update orders table as new order_items are added in

```sql
CREATE TRIGGER insert_new_orders
AFTER INSERT ON order_items
FOR EACH ROW
REPLACE INTO orders
SELECT
    order_id,
    MIN(created_at) AS created_at,
    MIN(website_session_id) as website_session_id,
    SUM(CASE WHEN is_primary_item = 1 THEN product_id ELSE NULL END) AS primary_product_id,
    COUNT(order_item_id) AS items_purchased,
    SUM(price_usd) AS price_usd,
    SUM(cogs_usd) AS cogs_usg
FROM order_items
WHERE order_id = new.order_id
GROUP BY order_id
ORDER BY order_id;
```

Website Sessions

Schema

```sql
CREATE TABLE website_sessions(
    website_session_id BIGINT NOT NULL,
    created_at DATETIME NOT NULL,
    user_id BIGINT NOT NULL,
    is_repeated_session INT NOT NULL,
    utm_source VARCHAR(30),
    utm_campaign VARCHAR(30),
    utm_content VARCHAR(30),
    device_type VARCHAR(30),
    http_referer VARCHAR(100),
    PRIMARY KEY(website_session_id)
);
```

```sql
CREATE TABLE website_pageviews(
    website_pageview_id BIGINT NOT NULL,
    created_at DATETIME NOT NULL,
    website_session_id BIGINT NOT NULL,
    pageview_url VARCHAR(100),
    PRIMARY KEY(website_pageviews_id),
    FOREIGN KEY(website_session_id) REFERENCES website_sessions(website_session_id)
);
```

View

```sql
CREATE VIEW monthly_sessions AS
SELECT
    YEAR(created_at) AS year,
    MONTH(created_at) AS month,
    utm_source,
    utm_campaign,
    COUNT(website_session_id) AS number_of_sessions
FROM website_sessions
GROUP BY 1,2,3,4;
```

Total Orders and Revenue – Stored Procedures

```sql
DELIMITER //
CREATE PROCEDURE order_performance (IN startDate DATE, IN endDate DATE)
BEGIN
    SELECT
    COUNT(order_id) AS total_orders,
    SUM(price_usd) AS Revenue
    FROM orders
    WHERE DATE(created_at) BETWEEN startDate AND endDate;
END //

DELIMITER ;
```

Simpler to run stored procedure for staff not familiar with SQL

```sql
CALL order_performance('2013-11-01', '2013-12-31');
```

	total_orders	Revenue
▶	1908	104893.62

Chat Support Data Schema

```
-- users
    -- user+id
    -- created_at
    -- first_name
    -- last_name

-- support_members
    -- support_member_id
    -- created_at
    -- first_name
    -- last_name

-- chat_sessions
    -- chat_session_id
    -- created_at
    -- chat_end_at
    -- user_id
    -- support_member_id
    -- website_session_id

-- chat_messages
    -- chat_message_id
    -- created_at
    -- chat_session_id
    -- user_id (null for support members)
    -- supprt_member_id (null for users)
```

```sql
CREATE TABLE users(
    user_id BIGINT NOT NULL,
    created_at DATETIME NOT NULL,
    first_name VARCHAR(30) NOT NULL,
    last_name VARCHAR(30) NOT NULL,
    PRIMARY KEY(user_id)
);

CREATE TABLE support_members(
    support_member_id BIGINT NOT NULL,
    created_at DATETIME NOT NULL,
    first_name VARCHAR(30) NOT NULL,
    last_name VARCHAR(30) NOT NULL,
    PRIMARY KEY(support_member_id)
);

CREATE TABLE chat_sessions(
    chat_session_id BIGINT NOT NULL,
    created_at DATETIME NOT NULL,
    chat_end_at DATETIME NOT NULL,
    user_id BIGINT NOT NULL,
    support_member_id BIGINT NOT NULL,
    website_session_id BIGINT NOT NULL,
    PRIMARY KEY(chat_session_id),
    FOREIGN KEY(user_id) REFERENCES users(user_id),
    FOREIGN KEY(support_member_id) REFERENCES support_members(support_member_id),
    FOREIGN KEY(website_session_id) REFERENCES website_sessions(website_session_id)
);

CREATE TABLE chat_messages(
    chat_message_id BIGINT NOT NULL,
    created_at DATETIME,
    chat_session_id BIGINT NOT NULL,
    user_id BIGINT DEFAULT NULL,
    support_member_id BIGINT DEFAULT NULL,
    FOREIGN KEY(chat_session_id) REFERENCES chat_sessions(chat_session_id),
    FOREIGN KEY(user_id) REFERENCES users(user_id),
    FOREIGN KEY(support_member_id) REFERENCES support_members(support_member_id)
);
```

Additional Stored Procedures and Views

Chat Report

```
DELIMITER //

CREATE PROCEDURE chats_handled(IN representative_id BIGINT, IN start_date DATE, IN end_date DATE)
BEGIN
    SELECT COUNT(*) AS Total_Chats
    FROM chat_sessions
    WHERE support_member_id = representative_id AND DATE(created_at) BETWEEN start_date AND end_date;
END //

DELIMITER ;
```

Order Volume and Revenue Report

```
CREATE VIEW monthly_volume AS
SELECT
    YEAR(created_at) AS year,
    MONTH(created_at) AS month,
    COUNT(order_id) AS Total_Orders,
    SUM(price_usd) AS Total_Revenue
FROM orders
GROUP BY 1, 2
ORDER BY 1, 2;
```

```
1 *   SELECT * FROM monthly_volume;
```

Result Grid | Filter Rows: | Ex

year	month	Total_Orders	Total_Revenue
2012	3	60	2999.40
2012	4	99	4949.01
2012	5	108	5398.92
2012	6	140	6998.60
2012	7	169	8448.31
2012	8	228	11397.72
2012	9	287	14347.13
2012	10	371	18546.29
2012	11	618	30893.82
2012	12	506	25294.94
2013	1	390	19966.10
2013	2	498	26515.02
2013	3	385	19896.15
2013	4	553	28584.47
2013	5	571	29364.29
2013	6	593	30544.07
2013	7	604	31143.96

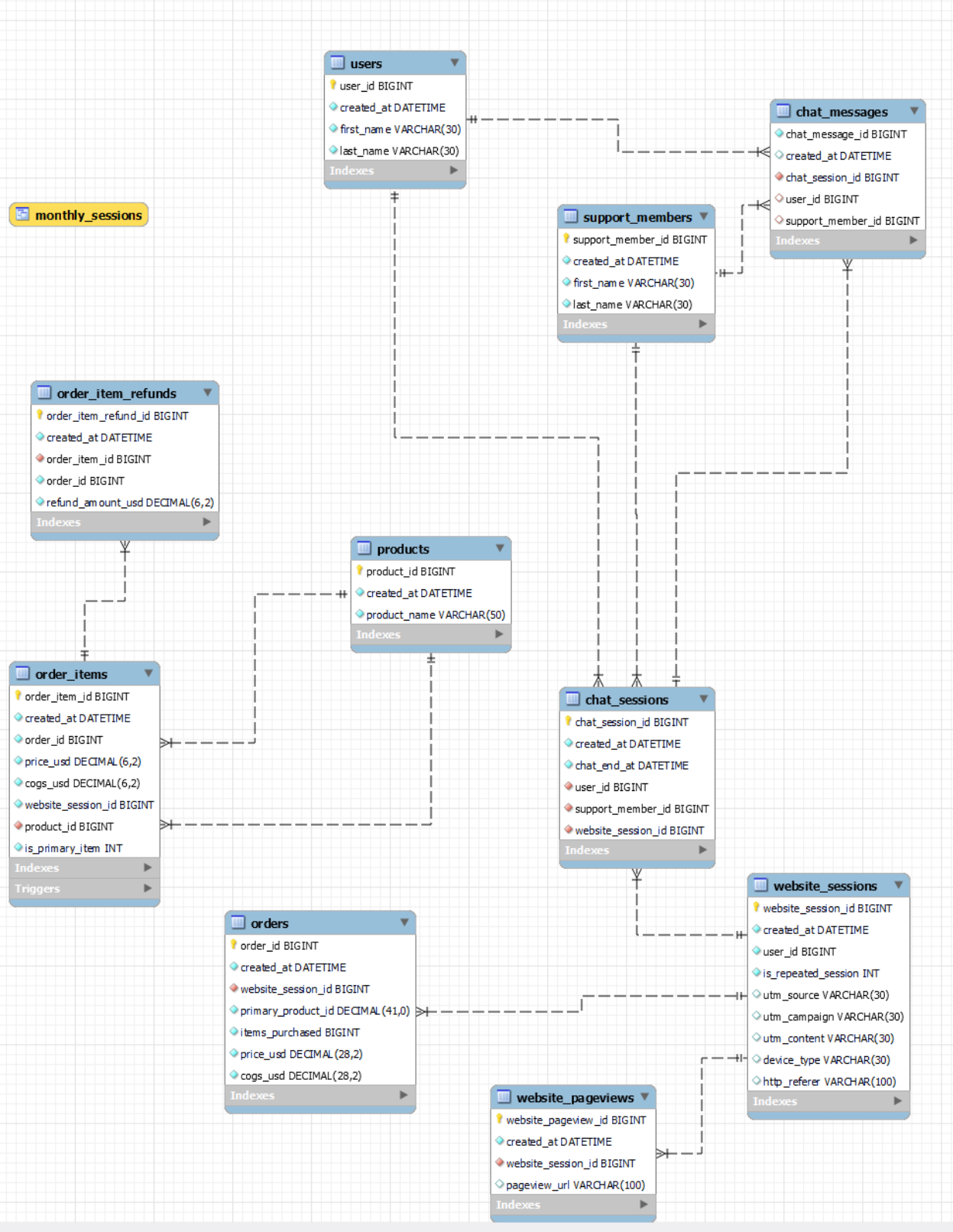
Monthly Website Traffic

```sql
CREATE VIEW monthly_website_traffic AS
SELECT
    YEAR(created_at) AS Year,
    MONTH(created_at) AS Month,
    COUNT(website_session_id) AS Total_traffic
FROM website_sessions
GROUP BY 1, 2
ORDER BY 1, 2;
```

```sql
SELECT * FROM monthly_website_traffic;
```

Year	Month	Total_traffic
2014	1	14825
2014	2	16285
2014	3	15669
2014	4	17353

Final EER Diagram



Security Plan Recommendation

Authentication

- Require strong passwords
- 8 Characters minimum, atleast one of the following:
    - Number
    - Upper and lower case letter
    - Special character
    - No sequentially repeating characters
- Recommend implementing two-factor or multi-factor authentication

Limit Access

- Make a list of people that will have access to sensitive information and try to cut it down.
- Only grant access based on job requirements, such as, read-only views, and only allowing select statements.
- Remove access immediately when employee leaves and reduce access when job requirements change.

Back-up

- Create regular logical and physical backups of db
- Keep backups offline and away from systems connected to internet.
    - On a separate offline computer, hard drives etc.
    - Prevents ransomware to also infect backup files.